

base

base-package

library(help = "base")

.bincode

.bincode(x, breaks, right = TRUE, include.lowest = FALSE)

x
breaks
right
include.lowest right = FALSE

cut.default(labels = FALSE)
[cut](#)breaksinclude.lowest = TRUEright = FALSE

x1NaNxNAbreaks

cuttabulate

```
## An example with non-unique breaks:
x <- c(0, 0.01, 0.5, 0.99, 1)
b <- c(0, 0, 1, 1)
.bincode(x, b, TRUE)
.bincode(x, b, FALSE)
.bincode(x, b, TRUE, TRUE)
.bincode(x, b, FALSE, TRUE)
```

.Device

.Devices[dev.cur](#).Device

.Device
.Devices

.Device
.Devices"null device""
"filepath"

.Machine

.Machine

.Machine

MACHAR"integer""double"
.Machine\$double.xmin5e-324

```

double.eps      x1 + x != 1double.base ^ ulp.digitsdouble.basedouble.rounding
                (double.base ^ double.ulp.digits) / 22.220446e-16
double.neg.eps  x1 - x != 1double.base ^ double.neg.ulp.digitsdouble.base
                double.rounding(double.base ^ double.neg.ulp.digits) /
                21.110223e-16double.neg.ulp.digits-(double.digits + 3)
                double.neg.eps
double.xmin     double.base ^ double.min.exp2.225074e-308
double.xmax     (1 - double.neg.eps) * double.base ^ double.max.exp1.797693e+308
double.base     2
double.digits   53
double.rounding

```

```

                    5
double.guard     double digitsdouble.base
                0
double.ulp.digits
                i1 + double.base ^ i != 1-(double.digits + 3)-52
double.neg.ulp.digits
                i1 - double.base ^ i != 1-(double.digits + 3)-53
double.exponent
                double.base11
double.min.exp  idouble.base ^ i-1022
double.max.exp  double.base1024
integer.max     231 - 1 = 2147483647
sizeof.long     long48
sizeof.longlong
                long long8
sizeof.longdouble
                long double1216long doubledouble
sizeof.pointer  SEXP48
sizeof.time_t   time_ttime_t8--with-internal-tzcode
longdouble.epslongdouble.neg.epslongdouble.digits...
                capabilities("long.double")"longdouble."long double"double.*"

```

```

capabilities("long.double")long doublesumx %% ylong doublereadBin
.Machinelongdouble.eps*.neg.eps*.digits*.rounding*.guard*.ulp.digits
*.neg.ulp.digits*.exponent*.min.exp*.max.expdouble.*
sizeof.longdouble
long doubledoublecapabilities("long.double").Machine"longdouble."

```

`.Platform`

```
.Machine
## or for a neat printout
noquote(unlist(format(.Machine)))
```

`.Platform`

`.Platform`

`.Platform`

```
OS.type      "unix""windows"
file.sep     "/"
dynlib.ext   ".dll""so""sl"dyn.loaddyn.load
GUI          "unknown"-g"X11""Tk""AQUA"R.app"Rgui""RTerm"
endian       "big""little"readBin
pkgType      options("pkgType")"source""mac.binary""win.binary"

path.sep     ":@"PATHTEXINPUTS
r_arch       ""
```

`.Platform$GUI"AQUA"R.app`

`/usr/local/binPATH`

`quartz`

`graphics = TRUE`[menuselect.list](#)

```
R.versionSys.infoR.version$platformosVersion
.Machinesystem
capabilitiesextSoftVersion
```

```
## Note: this can be done in a system-independent way by dir.exists()
if(.Platform$OS.type == "unix") {
  system.test <- function(...) system(paste("test", ...)) == 0L
  dir.exists2 <- function(dir)
    sapply(dir, function(d) system.test("-d", d))
  dir.exists2(c(R.home(), "/tmp", "~", "/NO")) # > T T T F
}
```

abbreviate

```
minlengthstrict = TRUE
```

```
abbreviate(names.arg, minlength = 4, use.classes = TRUE,
           dot = FALSE, strict = FALSE,
           method = c("left.kept", "both.sides"), named = TRUE)
```

```
names.arg      as.character
minlength
use.classes
dot            "."
strict         minlengthstrict = TRUE
method         "left.kept"
named          names
```

```
method = "left.kept"minlength
names.arg
NA
use.classesFALSE
```

```
names.argminlengthmethod = "both.sides"abbreviate()strict = FALSEminlength
names.arg
namesnames.argnames
Encoding
```


use.classes

substr

```
x <- c("abcd", "efgh", "abce")
abbreviate(x, 2)
abbreviate(x, 2, strict = TRUE) # >> 1st and 3rd are == "ab"

(st.abb <- abbreviate(state.name, 2))
stopifnot(identical(unname(st.abb),
  abbreviate(state.name, 2, named=FALSE)))
table(nchar(st.abb)) # out of 50, 3 need 4 letters :
as <- abbreviate(state.name, 3, strict = TRUE)
as[which(as == "Mss")]

## and without distinguishing vowels:
st.abb2 <- abbreviate(state.name, 2, FALSE)
cbind(st.abb, st.abb2)[st.abb2 != st.abb, ]

## method = "both.sides" helps: no 4-letters, and only 4 3-letters:
st.ab2 <- abbreviate(state.name, 2, method = "both")
table(nchar(st.ab2))
## Compare the two methods:
cbind(st.abb, st.ab2)
```

agrep

patternx

```
agrep(pattern, x, max.distance = 0.1, costs = NULL,
  ignore.case = FALSE, value = FALSE, fixed = TRUE,
  useBytes = FALSE)

agrep1(pattern, x, max.distance = 0.1, costs = NULL,
  ignore.case = FALSE, fixed = TRUE, useBytes = FALSE)
```

pattern	fixed = FALSE	as.character
x		as.character
max.distance		
	cost	
	all	
	insertions	

	deletions
	substitutions
	costallall
costs	insertionsdeletionssubstitutionsNULLas.integer
ignore.case	FALSETRUE
value	FALSETRUE
fixed	TRUE
useBytes	TRUE

```
trehttps://github.com/laurikari/tre
useBytes = TRUE"bytes"Encoding
```

```
agrepvalueTRUE
agrep1
```

```
xgrepadist
```

```
grepadistaregexec()
```

```
agrep("lasy", "1 lazy 2")
agrep("lasy", c(" 1 lazy 2", "1 lasy 2"), max.distance = list(sub = 0))
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max.distance = 2)
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max.distance = 2, value = TRUE)
agrep("laysy", c("1 lazy", "1", "1 LAZY"), max.distance = 2, ignore.case = TRUE)
```

```
all
```

```
all(..., na.rm = FALSE)
```

```
...
na.rm      NA
```

[Summary...](#)

```
x...NAna.rm = TRUE
TRUExTRUEFALSExFALSENAna.rm = FALSE...FALSENA
```

[Summary](#)x, ..., na.rm

```
all(logical(0))

all(all(x), all(y)) == all(x, y)

x
```

[anyallstopifnot](#)(*)all(*)

```
range(x <- sort(round(stats::rnorm(10) - 1.2, 1)))
if(all(x < 0)) cat("all x values are negative\n")

all(logical(0)) # true, as all zero of the elements are true.
```

all.equal

all.equal(x, y)xyall.equalifisTRUE(all.equal(...))[identical](#)

```
all.equal(target, current, ...)
```

```
## Default S3 method:
```

```
all.equal(target, current, ..., check.class = TRUE)
```

```
## S3 method for class 'numeric'
```

```
all.equal(target, current,
          tolerance = sqrt(.Machine$double.eps), scale = NULL,
          countEQ = FALSE,
          formatFUN = function(err, what) format(err),
```

```

        ..., check.attributes = TRUE, check.class = TRUE, giveErr = FALSE)

## S3 method for class 'character'
all.equal(target, current, ...,
          check.attributes = TRUE, check.class = TRUE)

## S3 method for class 'list'
all.equal(target, current, ...,
          check.attributes = TRUE, use.names = TRUE)

## S3 method for class 'environment'
all.equal(target, current, all.names = TRUE,
          evaluate = TRUE, ...)

## S3 method for class 'function'
all.equal(target, current, check.environment=TRUE, ...)

## S3 method for class 'POSIXt'
all.equal(target, current, ..., tolerance = 1e-3, scale,
          check.tzone = TRUE)

attr.all.equal(target, current, ...,
               check.attributes = TRUE, check.names = TRUE)

```

```

target
current      target
...
tolerance    ≥tolerance1.5e-8
scale        NULLlength(target)
countEQ      target == currentFALSEtargetcurrent
formatFUN    functionerrwhat
check.attributes
              attributestargetcurrent
check.class  data.class()targetcurrent
giveErr      logical"err"
use.names    list...
all.names    ls
evaluate     environmentlogicalnames
check.environment
              environment()check.environment=FALSEnls()
check.tzone  "tzone"targetcurrent
check.names  names(.)targetcurrent

```

```

all.equal(target, methods("all.equal"))
.....
scale = NULL; target; tolerance; targetNA; current; countEQ; NA
scale; scale
target; Mod; all.equal.numeric; tolerance; scale
list(target; current; is.vector; is.list)
environment; list; all.equal
all.equal.numeric("POSIXct"; tolerance; scale; target; current; check.tzone = FALSE)
attr.all.equal; attributes; NULL; character

TRUE; NULL; attr.all.equal; mode; "character"; target; current

=

identical; isTRUE; ==; all

all.equal(pi, 355/113)
# not precise enough (default tol) > relative error

quarts <- 1/4 + 1:10 # exact
d45 <- pi*quarts ; one <- rep(1, 10)
tan(d45) == one # mostly FALSE, as typically exact; embarrassingly,
tanpi(quarts) == one # (is always FALSE (Fedora 34; gcc 11.2.1))
stopifnot(all.equal(
  tan(d45), one)) # TRUE, but not if we are picky:
all.equal(tan(d45), one, tolerance = 0) # to see difference
all.equal(tan(d45), one, tolerance = 0, scale = 1) # "absolute diff.."
all.equal(tan(d45), one, tolerance = 0, scale = 1+(-2:2)/1e9) # "absolute"
all.equal(tan(d45), one, tolerance = 0, scale = 1+(-2:2)/1e6) # "scaled"

## advanced: equality of environments
ae <- all.equal(as.environment("package:stats"),
  asNamespace("stats"))
stopifnot(is.character(ae), length(ae) > 10,
  ## were incorrectly "considered equal" in R <= 3.1.1
  all.equal(asNamespace("stats"), asNamespace("stats")))

## A situation where 'countEQ = TRUE' makes sense:
x1 <- x2 <- (1:100)/10; x2[2] <- 1.1*x1[2]
## 99 out of 100 pairs (x1[i], x2[i]) are equal:
plot(x1, x2, main = "all.equal.numeric() -- not counting equal parts")
all.equal(x1, x2) ## "Mean relative difference: 0.1"
mtext(paste("all.equal(x1, x2) :", all.equal(x1, x2)), line = -2)
##' extract the 'Mean relative difference' as number:
all.eqNum <- function(...) as.numeric(sub(".*:", "", all.equal(...)))
set.seed(17)

```

```

## When x2 is jittered, typically all pairs (x1[i],x2[i]) do differ:
summary(r <- replicate(100, all.eqNum(x1, x2*(1+rnorm(x1)*1e-7))))
mtext(paste("mean(all.equal(x1, x2*(1 + eps_k))) {100 x} Mean rel.diff.=",
            signif(mean(r), 3)), line = -4, adj=0)
## With argument countEQ=TRUE, get "the same" (w/o need for jittering):
mtext(paste("all.equal(x1,x2, countEQ=TRUE) :",
            signif(all.eqNum(x1,x2, countEQ=TRUE), 3)), line= -6, col=2)

## Using giveErr=TRUE :
x1. <- x1 * (1+ 1e-9*rnorm(x1))
str(all.equal(x1, x1., giveErr=TRUE))
## logi TRUE
## - attr(*, "err")= num 8.66e-10
## - attr(*, "what")= chr "relative"

## Used with stopifnot(), still *showing* diff:
all.equalShow <- function (...) {
  r <- all.equal(..., giveErr=TRUE)
  cat(attr(r,"what"), "err:", attr(r,"err"), "\n")
  c(r) # can drop attributes, as not used anymore
}
# checks, showing error in any case:
stopifnot(all.equalShow(x1, x1.)) # -> relative err: 8.66002e-10
tryCatch(error=identity, stopifnot(all.equalShow(x1, 2*x1))) -> eAe
stopifnot(inherits(eAe, "error"))
# stopifnot(all.equal....()) giving smart msg:
cat(conditionMessage(eAe), "\n")

two <- structure(2, foo = 1, class = "bar")
all.equal(two^20, 2^20) # lots of diff
all.equal(two^20, 2^20, check.attributes = FALSE)# "target is bar, current is numeric"
all.equal(two^20, 2^20, check.attributes = FALSE, check.class = FALSE) # TRUE

## comparison of date-time objects
now <- Sys.time()
stopifnot(
  all.equal(now, now + 1e-4) # TRUE (default tolerance = 0.001 seconds)
)
all.equal(now, now + 0.2)
all.equal(now, as.POSIXlt(now, "UTC"))
stopifnot(
  all.equal(now, as.POSIXlt(now, "UTC"), check.tzone = FALSE) # TRUE
)

```

all.names

```
all.names(expr, functions = TRUE, max.names = -1L, unique = FALSE)
```

```
all.vars(expr, functions = FALSE, max.names = -1L, unique = TRUE)
```

```
expr
functions
max.names      -1
unique
```

substitute

```
all.names(expression(sin(x+y)))
all.names(quote(sin(x+y))) # or a call
all.vars(expression(sin(x+y)))
```

any

```
any(..., na.rm = FALSE)
```

```
...
na.rm      NA
```

Summary...

```
x...Nana.rm = TRUE
TRUExTRUEFALSExFALSENana.rm = FALSE...TRUENA
```

Summaryx, ..., na.rm

allany

```
range(x <- sort(round(stats::rnorm(10) - 1.2, 1)))
if(any(x < 0)) cat("x contains negative values\n")
```

aperm

```
aperm(a, perm, ...)
## Default S3 method:
aperm(a, perm = NULL, resize = TRUE, ...)
## S3 method for class 'table'
aperm(a, perm = NULL, resize = TRUE, keep.class = TRUE, ...)
```

```
a
perm      1:nnaanperm
resize     TRUE
keep.class a
...
```

```
apermresizeTRUEdimnamesresize = FALSEa
t
```

```
<J.C.Rougier@durham.ac.uk>
```

t

```
# interchange the first two subscripts on a 3-way array x
x <- array(1:24, 2:4)
xt <- aperm(x, c(2,1,3))
stopifnot(t(xt[,2]) == x[,2],
          t(xt[,3]) == x[,3],
          t(xt[,4]) == x[,4])
```

```
UCB <- aperm(UCBAdmissions, c(2,1,3))
UCB[1,,]
summary(UCB) # UCB is still a contingency table
```

append

```
append(x, values, after = length(x))
```

x

values

after

xvaluesx

```
append(1:5, 0:1, after = 3)
```

apply

```
apply(X, MARGIN, FUN, ..., simplify = TRUE)
```

X

MARGIN 12c(1, 2)X

FUN +%*%

... FUN

simplify

`Xdim``apply``as.matrix``as.array`

`FUN``match.fun``apply`

...`MARGIN``FUN`...`MARGIN``FUN``X``MARGIN``FUN`...

```

FUNnsimplifyTRUEapplyc(n, dim(X)[MARGIN])n > 1n1applyMARGINdim(X)[MARGIN]n0
FUNsimplifyFALSEapplyprod(dim(X)[MARGIN])dimMARGIN
as.vector

```

[lapplysimplify2arraytapplysweepaggregate](#)

```

## Compute row and column sums for a matrix:
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
dimnames(x)[[1]] <- letters[1:8]
apply(x, 2, mean, trim = .2)
col.sums <- apply(x, 2, sum)
row.sums <- apply(x, 1, sum)
rbind(cbind(x, Rtot = row.sums), Ctot = c(col.sums, sum(col.sums)))

stopifnot( apply(x, 2, is.vector))

## Sort the columns of a matrix
apply(x, 2, sort)

## keeping named dimnames
names(dimnames(x)) <- c("row", "col")
x3 <- array(x, dim = c(dim(x),3),
           dimnames = c(dimnames(x), list(C = paste0("cop.",1:3))))
identical(x, apply( x, 2, identity))
identical(x3, apply(x3, 2:3, identity))

##- function with extra args:
cave <- function(x, c1, c2) c(mean(x[c1]), mean(x[c2]))
apply(x, 1, cave, c1 = "x1", c2 = c("x1","x2"))

ma <- matrix(c(1:4, 1, 6:8), nrow = 2)
ma
apply(ma, 1, table) #--> a list of length 2
apply(ma, 1, stats::quantile) # 5 x n matrix with rownames

stopifnot(dim(ma) == dim(apply(ma, 1:2, sum)))

## Example with different lengths for each call
z <- array(1:24, dim = 2:4)
zseq <- apply(z, 1:2, function(x) seq_len(max(x)))
zseq      ## a 2 x 3 matrix
typeof(zseq) ## list
dim(zseq) ## 2 3
zseq[1,]
apply(z, 3, function(x) seq_len(max(x)))
# a list without a dim attribute

```

args

args(name)

name name

formals

NULL

NULL

NULL

formalshelpstr

```
## "regular" (non-primitive) functions "print their arguments"
## (by returning another function with NULL body which you also see):
args(ls)
args(graphics::plot.default)
utils::str(ls) # (just "prints": does not show a NULL)

## You can also pass a string naming a function.
args("scan")
## ...but :: package specification doesn't work in this case.
tryCatch(args("graphics::plot.default"), error = print)

## As explained above, args() gives a function with empty body:
list(is.f = is.function(args(scan)), body = body(args(scan)))

## Primitive functions mostly behave like non-primitive functions.
args(c)
args(`+`)
## primitive functions without well-defined argument list return NULL:
args(`if`)
```

Arithmetic

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %% y
x %/% y
```

```
xy
```

[OpsOps](#)

```
FALSETRUE
```

```
1 ^ yy ^ 01x ^ yInf-Inf
```

```
%%xy
```

```
%%x %/% yy1 %/% 0.21 %/% 0.250.20.24
```

```
(-8)^(1/3)NaNpow^man pow
```

```
+ -x
```

[warning](#)+ -*/^

```
%%x mod yr <- x %% y/%q <- x %/% y := floor(x/y)sign(r) == sign(y)
```

```
x == (x %% y) + y * (x %/% y)
```

```
y == 0%%NA_integer_NAtypeofInf - Inf
```

```
/^±(231 - 1)NA_integer_
```

[tsp](#)

[ArithOpsc](#)(e1, e2)e2

[.Machine](#)

```
-0.01/xInf-Infxidentical(0, -0, num.eq = FALSE)
```

``&`(x, y)`[Ops](#)

`**^`Deprecated

https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html

<https://www.iso.org/standard/57469.html>https://en.wikipedia.org/wiki/IEEE_754

https://en.wikipedia.org/wiki/Modulo_operation

[sqrtSpecial](#)

[Syntax](#)

[%*%](#)

```
x <- -1:12
x + 1
2 * x + 3
x %% 3 # is periodic 2 0 1 2 0 1 ...
x %% -3 # (ditto) -1 0 -2 -1 0 -2 ...
x %/% 5
x %% Inf # now is defined by limit (gave NaN in earlier versions of R)

## Illustrating PR#18677, see above
1 %/% print(0.2, digits=19)
```

array

```
array(data = NA, dim = length(data), dimnames = NULL)
as.array(x, ...)
is.array(x)
```

data [expressionas.vector](#)

dim

dimnames [NULLNULLdimNULL](#)

x

...

```

"dim" "dimnames"
matrix
str
"dim"
"dimnames" NULL "dim"
is.array
print.integer, 7

```

```

arraydimdimnamesdatadatadatadataNA0NULL
matrixarrayas.vectordata
as.arraydimdimnamesxnamesdim[names]
is.arrayTRUEFALSEdim

```

```

is.array

```

```

apermmatrixdimdimnames

```

```

dim(as.array(letters))
array(1:3, c(2,4)) # recycle 1:3 "2 2/3 times"
#      [,1] [,2] [,3] [,4]
#[1,]    1    3    2    1
#[2,]    2    1    3    2

```

```

array2DF

```

```

array2DFtapply

```

```

array2DF(x, responseName = "Value",
         sep = "", base = list(LETTERS),
         simplify = TRUE, allowLong = TRUE)

```

```

x
responseName
sep
base          x
simplify
allowLong     xxsimplify = TRUE

```

```

array2DFtapply
simplify = FALSEas.data.frame.tablex
xdimnamesbasesep
simplify = TRUE
xrbinddimnamesresponseName
xallowLong = TRUEresponseName
simplify2arrayresponseNamesep

```

```

length(dim(x)) + 1length(dim(x))xdimnamesx

```

```

tapplyas.data.frame.tablesplitaggregate

```

```

s1 <- with(ToothGrowth,
           tapply(len, list(dose, supp), mean, simplify = TRUE))

s2 <- with(ToothGrowth,
           tapply(len, list(dose, supp), mean, simplify = FALSE))

str(s1) # atomic array
str(s2) # list array

str(array2DF(s1, simplify = FALSE)) # Value column is vector
str(array2DF(s2, simplify = FALSE)) # Value column is list
str(array2DF(s2, simplify = TRUE)) # simplified to vector

### The remaining examples use the default 'simplify = TRUE'

## List array with list components: columns are lists (no simplification)

with(ToothGrowth,
     tapply(len, list(dose, supp),
           function(x) t.test(x)[c("p.value", "alternative")])) |>
  array2DF() |> str()

## List array with data frame components: columns are atomic (simplified)

with(ToothGrowth,
     tapply(len, list(dose, supp),
           function(x) with(t.test(x), data.frame(p.value, alternative)))) |>
  array2DF() |> str()

## named vectors

with(ToothGrowth,
     tapply(len, list(dose, supp),
           quantile)) |> array2DF()

## unnamed vectors: long format

```

```

with(ToothGrowth,
      tapply(len, list(dose, supp),
              sample, size = 5)) |> array2DF()

## unnamed vectors: wide format

with(ToothGrowth,
      tapply(len, list(dose, supp),
              sample, size = 5)) |> array2DF(allowLong = FALSE)

## unnamed vectors of unequal length

with(ToothGrowth[-1, ],
      tapply(len, list(dose, supp),
              sample, replace = TRUE)) |>
array2DF(allowLong = FALSE)

## unnamed vectors of unequal length with allowLong = TRUE
## (within-group bootstrap)

with(ToothGrowth[-1, ],
      tapply(len, list(dose, supp), sample, replace = TRUE)) |>
array2DF() |> str()

## data frame input

tapply(ToothGrowth, ~ dose + supp, FUN = with,
       data.frame(n = length(len), mean = mean(len), sd = sd(len))) |>
array2DF()

```

as.data.frame

```

as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'character'
as.data.frame(x, ...,
              stringsAsFactors = FALSE)

## S3 method for class 'list'
as.data.frame(x, row.names = NULL, optional = FALSE, ...,
              cut.names = FALSE, col.names = names(x), fix.empty.names = TRUE,
              new.names = !missing(col.names),
              check.names = !optional,
              stringsAsFactors = FALSE)

## S3 method for class 'matrix'
as.data.frame(x, row.names = NULL, optional = FALSE,

```



```

        make.names = TRUE, ...,
        stringsAsFactors = FALSE)

as.data.frame.vector(x, row.names = NULL, optional = FALSE, ...,
                     nm = deparse1(substitute(x)))

is.data.frame(x)

x
row.names      NULL
optional       TRUEmake.namesas.data.frame()optionaldata.frame(*, check.names =
!optional)make.namesmatrix
...
stringsAsFactors

cut.names      cut.names" ..."
col.names
fix.empty.names
              ""data.frame
new.names      col.namesNA
check.names    data.frame()
make.names     logicalFALSE,    NA,    TRUExTRUEmake.names(*,    unique=TRUE)
              make.names=NAFALSE
nm             character

as.data.frameas.data.frame.vector
as.data.frame.vectoris.atomicx
as.data.frame.as.data.frame.vector
as.data.frame"model.matrix""POSIXlt""POSIXct"

I
"data.frame"
row.names = NULLxI

as.data.frame""optional = TRUE
is.data.frameTRUE"data.frame"FALSE

data.framemake.namesas.data.frame.tabletable

```

```

L0 <- list(LETTERS[1:7], c(4L, 2:3, 5:7, 1L))
L <- L0; names(L) <- nms <- c("nam", "age")
d0 <- as.data.frame(L0, col.names = nms)
(d1 <- as.data.frame(L))
stopifnot(identical(d0, d1))

## showing possibilities on how NA names are handled:
L <- list(A = 1:4); names(L) <- NA
names(dL <- as.data.frame(L)) # "NA."
names(dL1 <- as.data.frame(L, col.names = names(L))) # "NA."
##
names(dL1 <- as.data.frame(L, check.names=FALSE)) # "NA"
##
names(dL2 <- as.data.frame(L, col.names = names(L), check.names=FALSE)) # NA
names(dLn <- as.data.frame(L, new.names = TRUE, check.names=FALSE)) # NA

```

as.Date

"Date"

```

as.Date(x, ...)
## S3 method for class 'character'
as.Date(x, format, tryFormats = c("%Y-%m-%d", "%Y/%m/%d"),
        optional = FALSE, ...)
## S3 method for class 'numeric'
as.Date(x, origin, ...)
## S3 method for class 'POSIXct'
as.Date(x, tz = "UTC", ...)

## S3 method for class 'Date'
format(x, format = "%Y-%m-%d", ...)

## S3 method for class 'Date'
as.character(x, ...)

```

```

x
format      character tryFormats NA strptime()
tryFormats  character format format
optional    logical NA
origin      Date as.Date(origin, ...) missing "1970-01-01"
tz
...

```

xformat

as.DateNA"POSIXlt""POSIXct""date""dates"

as.Dateorigin

formatas.character

formatas.characterNANA_character_

as.Date"Date"

"2001-02-03"

NAglibc

<https://www.qsl.net/g1smd/isopdf.htm>

locales

strftimestrptimestrptimeglibc

```
## locale-specific version of the date
format(Sys.Date(), "%a %b %d")
```

```
## read in date info in format 'ddmmmyyyy'
## This will give NA(s) in some locales; setting the C locale
## as in the commented lines will overcome this on most systems.
## lct <- Sys.getlocale("LC_TIME"); Sys.setlocale("LC_TIME", "C")
x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
z <- as.Date(x, "%d%b%Y")
## Sys.setlocale("LC_TIME", lct)
z
```

```
## read in date/time info in format 'm/d/y'
dates <- c("02/27/92", "02/27/92", "01/14/92", "02/28/92", "02/01/92")
as.Date(dates, "%m/%d/%y")
```

```
## date given as number of days since 1900-01-01 (a date in 1989)
as.Date(32768, origin = "1900-01-01")
```

```

## Excel is said to use 1900-01-01 as day 1 (Windows default) or
## 1904-01-01 as day 0 (Mac default), but this is complicated by Excel
## incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel
as.Date(35981, origin = "1899-12-30") # 1998-07-05
## and Mac Excel
as.Date(34519, origin = "1904-01-01") # 1998-07-05
## (these values come from http://support.microsoft.com/kb/214330)

## Experiment shows that Matlab's origin is 719529 days before ours,
## (it takes the non-existent 0000-01-01 as day 1)
## so Matlab day 734373 can be imported as
as.Date(734373) - 719529 # 2010-08-23
## (value from
## http://www.mathworks.de/de/help/matlab/matlab\_prog/represent-date-and-times-in-MATLAB.html)

## Time zone effect
z <- ISOdate(2010, 04, 13, c(0,12)) # midnight and midday UTC
as.Date(z) # in UTC
## these time zone names are common
as.Date(z, tz = "NZ")
as.Date(z, tz = "HST") # Hawaii

```

as.environment

[environment](#)

as.environment(x)

```

x                -1
                  list2env(x, parent = emptyenv())
                  is.object(x)classas.environment

```

[environmentsearchlist2env](#)

```

as.environment(1) ## the global environment
identical(globalenv(), as.environment(1)) ## is TRUE
try( ## <- stats need not be attached
    as.environment("package:stats"))
ee <- as.environment(list(a = "A", b = pi, ch = letters[1:8]))
ls(ee) # names of objects in ee
utils::ls.str(ee)

```

as.function

```
as.function
as.function.defaultx"call"

as.function(x, ...)

## Default S3 method:
as.function(x, envir = parent.frame(), ...)
```

x

...

envir

functionalist

```
as.function(alist(a = , b = 2, a+b))
as.function(alist(a = , b = 2, a+b))(3)
```

as.POSIX*

```
"POSIXlt""POSIXct"

as.POSIXct(x, tz = "", ...)
as.POSIXlt(x, tz = "", ...)

## S3 method for class 'character'
as.POSIXlt(x, tz = "", format,
  tryFormats = c("%Y-%m-%d %H:%M:%OS",
    "%Y/%m/%d %H:%M:%OS",
    "%Y-%m-%d %H:%M",
    "%Y/%m/%d %H:%M",
    "%Y-%m-%d",
```

```

        "%Y/%m/%d"),
        optional = FALSE, ...)
## Default S3 method:
as.POSIXlt(x, tz = "",
           optional = FALSE, ...)
## S3 method for class 'numeric'
as.POSIXlt(x, tz = "", origin, ...)

## S3 method for class 'Date'
as.POSIXct(x, tz = "UTC", ...)
## S3 method for class 'Date'
as.POSIXlt(x, tz = "UTC", ...)
## S3 method for class 'numeric'
as.POSIXct(x, tz = "", origin, ...)

## S3 method for class 'POSIXlt'
as.double(x, ...)

x
tz          "" "GMT"
...
format      strptime
tryFormats  character formatformat
optional    logical NA
origin      as.POSIXct(tz = "GMT") "1970-01-01"

as.POSIX*"Date"
"2001-02-03""2001/02/03""14:52""14:52:03""01/02/03"strptimeformatNA
formatLC_TIME Sys.setlocale%a%A%b%B%p
NA

"POSIXlt"strptime"POSIXct""POSIXlt""POSIXct"

as.POSIXct(strptime("2011-03-27 01:30:00", "%Y-%m-%d %H:%M:%S"))
as.POSIXct(strptime("2010-10-31 01:30:00", "%Y-%m-%d %H:%M:%S"))

NAstrftime

as.POSIXctas.POSIXltPOSIXt
tzas.POSIXlt"tzone"NA

"POSIXct"
"POSIXlt""POSIXct"
"POSIXlt"format

```

strptime

Sys.timezone

```
(z <- Sys.time())          # the current datetime, as class "POSIXct"
unclass(z)                 # a large integer
floor(unclass(z)/86400)    # the number of days since 1970-01-01 (UTC)
(now <- as.POSIXlt(Sys.time())) # the current datetime, as class "POSIXlt"
str(unclass(now))          # the internal list ; use now$hour, etc :
now$year + 1900            # see ?DateTimeClasses
months(now); weekdays(now) # see ?months; using LC_TIME locale

## suppose we have a time in seconds since 1960-01-01 00:00:00 GMT
## (the origin used by SAS)
z <- 1472562988
# ways to convert this
as.POSIXct(z, origin = "1960-01-01")      # local
as.POSIXct(z, origin = "1960-01-01", tz = "GMT") # in UTC

## SPSS dates (R-help 2006-02-16)
z <- c(10485849600, 10477641600, 10561104000, 10562745600)
as.Date(as.POSIXct(z, origin = "1582-10-14", tz = "GMT"))

## Stata date-times: milliseconds since 1960-01-01 00:00:00 GMT
## format %tc excludes leap-seconds, assumed here
## For format %tC including leap seconds, see foreign::read.dta()
z <- 1579598122120
op <- options(digits.secs = 3)
# avoid rounding down: milliseconds are not exactly representable
as.POSIXct((z+0.1)/1000, origin = "1960-01-01")
options(op)

## Matlab 'serial day number' (days and fractional days)
z <- 7.343736909722223e5 # 2010-08-23 16:35:00
as.POSIXct((z - 719529)*86400, origin = "1970-01-01", tz = "UTC")

as.POSIXlt(Sys.time(), "GMT") # the current time in UTC

## These may not be correct names on your system
as.POSIXlt(Sys.time(), "America/New_York") # in New York
as.POSIXlt(Sys.time(), "EST5EDT")         # alternative.
as.POSIXlt(Sys.time(), "EST" )            # somewhere in Eastern Canada
as.POSIXlt(Sys.time(), "HST")             # in Hawaii
as.POSIXlt(Sys.time(), "Australia/Darwin")

tab <- file.path(R.home("share"), "zoneinfo", "zone1970.tab")
if(file.exists(tab)) { # typically on Windows; on Linux when `configure --with-internal-tzcode`
  cols <- c("code", "coordinates", "TZ", "comments")
  tmp <- read.delim(tab,
                    header = FALSE, comment.char = "#", col.names = cols)
  if(interactive()) View(tmp)
  head(tmp, 10)
}
```

AsIs

I(x)

x

I

```
data.frameI()data.framestringsAsFactors = TRUEIas.data.frame
"AsIs""AsIs"[as.data.frameprintformat
formula"+"-"*""^"terms.formula
```

"AsIs"

data.frameformula

asplit

asplit(x, MARGIN, drop = FALSE)

x

MARGIN 12c(1, 2)x

drop

apply(x, MARGIN, identity, simplify = FALSE)split(x, slice.index(x, MARGIN))

*dvd*edrop*dvd*exMARGIN


```
## A 3-dimensional array of dimension 2 x 3 x 4:
d <- 2 : 4
x <- array(seq_len(prod(d)), d)
x
## Splitting by margin 2 gives a 1-d list array of length 3
## consisting of 2 x 4 arrays:
asplit(x, 2)
## Splitting by margins 1 and 2 gives a 2 x 3 list array
## consisting of 1-d arrays of length 4:
asplit(x, c(1, 2))
## Compare to
split(x, slice.index(x, c(1, 2)))

## A 2 x 3 matrix:
(x <- matrix(1 : 6, 2, 3))
## To split x by its rows, one can use
asplit(x, 1)
## or less efficiently
split(x, slice.index(x, 1))
split(x, row(x))
```

assign

```
assign(x, value, pos = -1, envir = as.environment(pos),
       inherits = FALSE, immediate = TRUE)
```

```
x
value      x
pos
envir      environment
inherits
immediate
```

```
xmake.names
pos-1searchenvironmentsys.frameenvir
assign
attachwith
```

```
valuexenvir
inheritsTRUExlockBinding
inheritsFALSEenvir
```

`<-getassign()existsenvironment`

```
for(i in 1:6) { ##-- Create objects 'r.1', 'r.2', ... 'r.6' --
  nam <- paste("r", i, sep = ".")
  assign(nam, 1:i)
}
ls(pattern = "^r..$")

##-- Global assignment within a function:
myf <- function(x) {
  innerf <- function(x) assign("Global.res", x^2, envir = .GlobalEnv)
  innerf(x+1)
}
myf(3)
Global.res # 16

a <- 1:4
assign("a[1]", 2)
a[1] == 2      # FALSE
get("a[1]") == 2  # TRUE
```

assignOps

```
x <- value
x <<- value
value -> x
value ->> x
```

```
x = value
```

```
x
value      x
```

```
<-=<-=
```

```
<<-->>
```

```
xz[[1]]
```

```
<- = <<-
```

```
valuea <- b <- c <- 6
```

```
=
```

```
assigngetx[i] <- v[<-environment
```

```
attach
```

```
attach(what, pos = 2L, name = deparse1(substitute(what), backtick=FALSE),  
       warn.conflicts = TRUE)
```

```
what          data.frame list save NULL  
pos           search()  
name          package:library  
warn.conflicts TRUE message() conflicts.conflicts.OK  
              warn.conflicts warning() message()
```

```
search  
heightwomen$height  
pospos = 1  
<<-assignattach  
what = NULL assignload sys.source  
"package:"library file:namesearch as.environment
```

```
environment "name"
```

```
attachdetach  
withattachdetach what save() attach() load()  
with  
    name  
    attach on.exit detach  
attach
```

[librarydetachsearchobjectsenvironmentwith](#)

```
require(utils)

summary(women$height) # refers to variable 'height' in the data frame
attach(women)
summary(height)       # The same variable now available by name
height <- height*2.54 # Don't do this. It creates a new variable
                     # in the user's workspace

find("height")
summary(height)       # The new variable in the workspace
rm(height)
summary(height)       # The original variable.
height <-<- height*25.4 # Change the copy in the attached environment
find("height")
summary(height)       # The changed copy
detach("women")
summary(women$height) # unchanged

## Not run: ## create an environment on the search path and populate it
sys.source("myfuns.R", envir = attach(NULL, name = "myfuns"))

## End(Not run)
```

attr

```
attr(x, which, exact = FALSE)
attr(x, which) <- value
```

```
x
which
exact      which
value      NULL
```

[NULLsymbolnameprimitive](#)

`whichxexact = TRUEoptions(warnPartialMatchAttr = TRUE)`

[classcommentdimdimnamesnamesrow.namesstsplevels](#)levels

which

[NULL](#)attr

NULL

attributes

```
# create a 2 by 5 matrix
x <- 1:10
attr(x,"dim") <- c(2, 5)

S <- sum
attr(S, "foo") <- NA # gives a warning, will become an error!
attributes(sum) <- NULL # revert to sanity
```

attributes

```
attributes(x)
attributes(x) <- value
mostattributes(x) <- value
```

```
x          symbolnameprimitive
value      listNULL
```

```
attrNULLlist
classcommentdimdimnamesnamesrow.nameslevelslevels
attributes()identical()attrib.as.set = TRUE"NA"
dimdimdimnames
mostattributesdimnamesdimnamesattributesrow.names
attributes
NULL
attributes
```

attrstructure

```

x <- cbind(a = 1:3, pi = pi) # simple matrix with dimnames
attributes(x)

## strip an object's attributes:
attributes(x) <- NULL
x # now just a vector of length 6

mostattributes(x) <- list(mycomment = "really special", dim = 3:2,
  dimnames = list(LETTERS[1:3], letters[1:5]), names = paste(1:6))
x # dim(), but not {dim}names

```

autoload

```

autoloadautoloadername.AutoloadEnvnameautoloadernamepackage
.Autoloaded

```

```

autoload(name, package, reset = FALSE, ...)
autoloader(name, package, ...)

```

```

.AutoloadEnv
.Autoloaded

```

```

name
package
reset          autoloader
...            library

```

[delayedAssignlibrary](#)

```

require(stats)
autoload("interpSpline", "splines")
search()
ls("Autoloads")
.Autoloaded

```

```

x <- sort(stats::rnorm(12))
y <- x^2
is <- interpSpline(x, y)
search() ## now has splines
detach("package:splines")
search()
is2 <- interpSpline(x, y+x)
search() ## and again
detach("package:splines")

```

backsolve

```
backsolve(r, x, k = ncol(r), upper.tri = TRUE,  
          transpose = FALSE)  
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE,  
            transpose = FALSE)
```

```
rl  
x  
k          rx  
upper.tri  TRUEr  
transpose  TRUEr' * y = xyt(r) %*% y == x
```

```
x <- backsolve (R, b)  $Rx = b$   
x <- forwardsolve(L, b)  $Lx = b$   
rlkxk  
dtrsm
```

```
xx
```

cholqrsolve

```
## upper triangular matrix 'r':  
r <- rbind(c(1,2,3),  
           c(0,1,1),  
           c(0,0,2))  
( y <- backsolve(r, x <- c(8,4,2)) ) # -1 3 1  
r %*% y # == x = (8,4,2)  
backsolve(r, x, transpose = TRUE) # 8 -12 -5
```

balancePOSIXlt

```
"POSIXlt"
unCfillPOSIXlt(x)balancePOSIXlt(x,          fill.only=TRUE,          classed=FALSE)
unclass(balancePOSIXlt(x, fill.only=TRUE))
```

```
balancePOSIXlt(x, fill.only = FALSE, classed = TRUE)
unCfillPOSIXlt(x)
```

```
x          "POSIXlt"POSIXlt
fill.only  logicalbalancePOSIXlt(x, ..)x$wdayx$yday
classed    logicalbalancePOSIXlt(x,          classed          =          FALSE)
unclass(balancePOSIXlt(x))
```

```
"POSIXlt"xlength"POSIXct"length()"Date"[[<-
secminhourmdaymon
balancePOSIXlt(x)"POSIXlt"xas.POSIXltfill.only = TRUEx
balancePOSIXlt()POSIXltlogical"balanced"NATRUE
```

```
POSIXltas.POSIXlt
```

```
## FIXME: this should also work for regular (non-UTC) time zones.
TZ <- "UTC"
# Could be
# d1 <- as.POSIXlt("2000-01-02 3:45", tz = TZ)
# on systems (almost all) which have tm_zone.
oldTZ <- Sys.getenv('TZ', unset = "unset")
Sys.setenv(TZ = "UTC")
d1 <- as.POSIXlt("2000-01-02 3:45")
d1$min <- d1$min + (0:16)*20L
(f1 <- format(d1))
str(unclass(d1))      # only $min is of length > 1
df <- balancePOSIXlt(d1, fill.only = TRUE) # a "POSIXlt" object
str(unclass(df))      # all of length 17; 'min' unchanged
db <- balancePOSIXlt(d1, classed = FALSE) # a list
stopifnot(identical(
  unCfillPOSIXlt(d1),
  balancePOSIXlt(d1, fill.only = TRUE, classed = FALSE)))
str(db) # of length 17 *and* in range

if(oldTZ == "unset") Sys.unsetenv('TZ') else Sys.setenv(TZ = oldTZ)
```

basename

basename
dirnamepath"."

basename(path)
dirname(path)

path

dirname

path
dirname"."
pathNA
""

\\dirname\\

"/".

[file.pathpath.expand](#)

basename(file.path("", "p1", "p2", "p3", c("file1", "file2")))
dirname (file.path("", "p1", "p2", "p3", "filename"))

Bessel

$J_\nu Y_\nu I_\nu K_\nu$

```
besseli(x, nu, expon.scaled = FALSE)
besselk(x, nu, expon.scaled = FALSE)
besselj(x, nu)
bessely(x, nu)
```

x ≥ 0
nu
expon.scaled TRUE $I_\nu K_\nu$

expon.scaled = TRUE $e^{-x} I_\nu(x) e^x K_\nu(x)$
 $\nu < 0$ besselKnu
nu

expon.scaled = TRUE

<maechler@stat.math.ethz.ch>

<https://netlib.org/specfun/ribesl../rjbesl>

besseli
besselj besseli
besselk
bessely $Y_\nu(x)$

$\Gamma(x)B(a,b)$

```
require(graphics)

nus <- c(0:5, 10, 20)

x <- seq(0, 4, length.out = 501)
plot(x, x, ylim = c(0, 6), ylab = "", type = "n",
     main = "Bessel Functions I_nu(x)")
for(nu in nus) lines(x, besselI(x, nu = nu), col = nu + 2)
legend(0, 6, legend = paste("nu=", nus), col = nus + 2, lwd = 1)

x <- seq(0, 40, length.out = 801); y1 <- c(-.5, 1)
plot(x, x, ylim = y1, ylab = "", type = "n",
     main = "Bessel Functions J_nu(x)")
abline(h=0, v=0, lty=3)
for(nu in nus) lines(x, besselJ(x, nu = nu), col = nu + 2)
legend("topright", legend = paste("nu=", nus), col = nus + 2, lwd = 1, bty="n")

## Negative nu's -----
xx <- 2:7
nu <- seq(-10, 9, length.out = 2001)
## --- I() ---
matplot(nu, t(outer(xx, nu, besselI)), type = "l", ylim = c(-50, 200),
     main = expression(paste("Bessel ", I[nu](x), " for fixed ", x,
                             ", as ", f(nu))),
     xlab = expression(nu))
abline(v = 0, col = "light gray", lty = 3)
legend(5, 200, legend = paste("x=", xx), col=seq(xx), lty=1:5)

## --- J() ---
bJ <- t(outer(xx, nu, besselJ))
matplot(nu, bJ, type = "l", ylim = c(-500, 200),
     xlab = quote(nu), ylab = quote(J[nu](x)),
     main = expression(paste("Bessel ", J[nu](x), " for fixed ", x)))
abline(v = 0, col = "light gray", lty = 3)
legend("topright", legend = paste("x=", xx), col=seq(xx), lty=1:5)

## ZOOM into right part:
matplot(nu[nu > -2], bJ[nu > -2,], type = "l",
     xlab = quote(nu), ylab = quote(J[nu](x)),
     main = expression(paste("Bessel ", J[nu](x), " for fixed ", x)))
abline(h=0, v = 0, col = "gray60", lty = 3)
legend("topright", legend = paste("x=", xx), col=seq(xx), lty=1:5)

##----- x --> 0 -----
x0 <- 2^seq(-16, 5, length.out=256)
plot(range(x0), c(1e-40, 1), log = "xy", xlab = "x", ylab = "", type = "n",
     main = "Bessel Functions J_nu(x) near 0\n log - log scale") ; axis(2, at=1)
for(nu in sort(c(nus, nus+0.5)))
  lines(x0, besselJ(x0, nu = nu), col = nu + 2, lty= 1+ (nu%1 > 0))
legend("right", legend = paste("nu=", paste(nus, nus+0.5, sep=", ")),
     col = nus + 2, lwd = 1, bty="n")
```

```

x0 <- 2^seq(-10, 8, length.out=256)
plot(range(x0), 10^c(-100, 80), log = "xy", xlab = "x", ylab = "", type = "n",
      main = "Bessel Functions K_nu(x) near 0\n log - log scale") ; axis(2, at=1)
for(nu in sort(c(nus, nus+0.5)))
  lines(x0, besselK(x0, nu = nu), col = nu + 2, lty= 1+ (nu%%1 > 0))
legend("topright", legend = paste("nu=", paste(nus, nus + 0.5, sep = ", ")),
      col = nus + 2, lwd = 1, bty="n")

x <- x[x > 0]
plot(x, x, ylim = c(1e-18, 1e11), log = "y", ylab = "", type = "n",
      main = "Bessel Functions K_nu(x)"); axis(2, at=1)
for(nu in nus) lines(x, besselK(x, nu = nu), col = nu + 2)
legend(0, 1e-5, legend=paste("nu=", nus), col = nus + 2, lwd = 1)

yl <- c(-1.6, .6)
plot(x, x, ylim = yl, ylab = "", type = "n",
      main = "Bessel Functions Y_nu(x)")
for(nu in nus){
  xx <- x[x > .6*nus]
  lines(xx, besselY(xx, nu=nus), col = nus+2)
}
legend(25, -.5, legend = paste("nu=", nus), col = nus+2, lwd = 1)

## negative nu in bessel_Y -- was bogus for a long time
curve(besselY(x, -0.1), 0, 10, ylim = c(-3,1), ylab = "")
for(nu in c(seq(-0.2, -2, by = -0.1)))
  curve(besselY(x, nu), add = TRUE)
title(expression(besselY(x, nu) * " " " *
  {nu == list(-0.1, -0.2, ..., -2)}))

```

bindenv

```

lockEnvironment(env, bindings = FALSE)
environmentIsLocked(env)
lockBinding(sym, env)
unlockBinding(sym, env)
bindingIsLocked(sym, env)

makeActiveBinding(sym, fun, env)
bindingIsActive(sym, env)
activeBindingFunction(sym, env)

```

```

env
bindings
sym
fun

```

```
lockEnvironment
lockBinding
makeActiveBindingfunenvsymfunsetRefClass.onLoad
```

```
bindingIsLockedenvironmentIsLockedNULL
```

```
# locking environments
e <- new.env()
assign("x", 1, envir = e)
get("x", envir = e)
lockEnvironment(e)
get("x", envir = e)
assign("x", 2, envir = e)
try(assign("y", 2, envir = e)) # error

# locking bindings
e <- new.env()
assign("x", 1, envir = e)
get("x", envir = e)
lockBinding("x", e)
try(assign("x", 2, envir = e)) # error
unlockBinding("x", e)
assign("x", 2, envir = e)
get("x", envir = e)

# active bindings
f <- local( {
  x <- 1
  function(v) {
    if (missing(v))
      cat("get\n")
    else {
      cat("set\n")
      x <- v
    }
  }
  x
})
makeActiveBinding("fred", f, .GlobalEnv)
bindingIsActive("fred", .GlobalEnv)
fred
fred <- 2
fred
```

bitwise

```
bitwNot(a)
bitwAnd(a, b)
bitwOr(a, b)
bitwXor(a, b)
```

```
bitwShiftL(a, n)
bitwShiftR(a, n)
```

```
ab
n
```

```
NA
```

```
NANA
```

```
!&|xorraw
"octmode""hexmode"
231
```

```
bitwNot(0:12) # -1 -2 ... -13
bitwAnd(15L, 7L) # 7
bitwOr (15L, 7L) # 15
bitwXor(15L, 7L) # 8
bitwXor(-1L, 1L) # -2
```

```
## The "same" for 'raw' instead of integer :
rr12 <- as.raw(0:12) ; rbind(rr12, !rr12)
c(r15 <- as.raw(15), r7 <- as.raw(7)) # 0f 07
r15 & r7      # 07
r15 | r7      # 0f
xor(r15, r7) # 08
```

```
bitwShiftR(-1, 1:31) # shifts of 232-1 = 4294967295
```

body

formals

```
body(fun = sys.function(sys.parent()))  
body(fun, envir = environment(fun)) <- value
```

fun

envir

value

funbody

{

body{[symbol](#)pi3"R"

[environment](#)[attributes](#)value"expression"

[formals](#)[body](#)[environment](#)

[alist](#)[args](#)[function](#)

```
body(body)  
f <- function(x) x^5  
body(f) <- quote(5^x)  
## or equivalently body(f) <- expression(5^x)  
f(3) # = 125  
body(f)
```

```
## creating a multi-expression body  
e <- expression(y <- x^2, return(y)) # or a list  
body(f) <- as.call(c(as.name("{"), e))  
f  
f(8)  
## Using substitute() may be simpler than 'as.call(c(as.name("{",...)))':  
stopifnot(identical(body(f), substitute({ y <- x^2; return(y) })))
```

bquote

```
bquote().wheresplice = TRUE..()
```

```
bquote(expr, where = parent.frame(), splice = FALSE)
```

```
expr
```

```
where
```

```
splice          TRUE
```

quotesubstitute

```
require(graphics)
```

```
a <- 2
```

```
bquote(a == a)
```

```
quote(a == a)
```

```
bquote(a == .(a))
```

```
substitute(a == A, list(A = a))
```

```
plot(1:10, a*(1:10), main = bquote(a == .(a)))
```

```
## to set a function default arg
```

```
default <- 1
```

```
bquote( function(x, y = .(default)) x+y )
```

```
exprs <- expression(x <- 1, y <- 2, x + y)
```

```
bquote(function() {..(exprs)}), splice = TRUE)
```

browser

```
browser
```

```
browser(text = "", condition = NULL, expr = TRUE, skipCalls = 0L)
```



```
text
condition
expr          logical
skipCalls
```

```
browser
textconditionbrowserTextbrowserCondition
exprexprFALSEif
skipCallsbrowser()
```

```
c
cont c
f
help
n browsernc
s sc
where
r "resume""resume"
Q
"browserNLdisabled"TRUEns
ls()n(n)
options(deparse.max.lines)
Browse[]>debug21debugger
```

```
repeat withRestarts(
  withCallingHandlers(
    readEvalPrint(),
    error = function(cnd) {
      cat("Error:", conditionMessage(cnd), "\n")
      invokeRestart("browser")
    }
  ),
  browser = function(...) NULL
)

readEvalPrint <- function(env = parent.frame()) {
  print(eval(parse(prompt = "Browse[n]> "), env))
}
```

```
"warning"
```

[debugtracebackbrowserText](#)

browserText

```
browserText(n = 1)
browserCondition(n = 1)
browserSetDebug(n = 1)
```

n

```
browserbrowserTextbrowserCondition
browserSetDebugsys.framebrowserSetDebugbrowserSetDebugnc
```

```
browserTextbrowserCondition
browserSetDebug
```

[browser](#)

builtins

```
builtins(internal = FALSE)
```

```
internal      .Internal
```

```
builtins()ls(baseenv(), all.names = TRUE)  
builtins(TRUE).Internal((args ...))
```

by

```
bytapply
```

```
by(data, INDICES, FUN, ..., simplify = TRUE)
```

```
data  
INDICES      nrow(data)INDICESfsplit  
FUN          functiondata  
...          FUN  
simplify     tapply
```

```
split()FUN  
FUN
```

```
"by"simplifytapply
```

```
tapplysimplify2arrayarray2DFave
```

```

require(stats)
by(warbreaks[, 1:2], warbreaks[, "tension"], summary)
by(warbreaks[, 1], warbreaks[, -1], summary)
by(warbreaks, warbreaks[, "tension"],
  function(x) lm(breaks ~ wool, data = x))

## now suppose we want to extract the coefficients by group
tmp1 <- with(warbreaks,
  by(warbreaks, tension,
    function(x) lm(breaks ~ wool, data = x)))
sapply(tmp1, coef)

## another way
by(warbreaks, ~ tension,
  with, coef(lm(breaks ~ wool))) |> array2DF(simplify = TRUE)

```

c

```

## S3 Generic function
c(...)

## Default S3 method:
c(..., recursive = FALSE, use.names = TRUE)

```

```

...          NULL
recursive    recursive = TRUE
use.names    names

namesymbolcalllistrecursive = TRUE
complexNANA_complex_NA
c.factor
carrayas.vectorc

```

NULLNULL

(x, ...)

unlistas.vector

```
c(1, 7:9)
c(1:5, 10.5, "next")

## uses with a single argument to drop attributes
x <- 1:4
names(x) <- letters[1:4]
x
c(x)          # has names
as.vector(x)  # no names
dim(x) <- c(2,2)
x
c(x)
as.vector(x)

## append to a list:
ll <- list(A = 1, c = "C")
## do *not* use
c(ll, d = 1:3) # which is == c(ll, as.list(c(d = 1:3)))
## but rather
c(ll, d = list(1:3)) # c() combining two lists

## descend through lists:
c(list(A = c(B = 1)), recursive = TRUE)
c(list(A = c(B = 1, C = 2), B = c(E = 7)), recursive = TRUE)
```

call

`mode"call"("`

```
call(name, ...)
is.call(x)
as.call(x)
```

name

...

x

```

call name...
  callname
is.call xmode"call"("
  is.call(x)typeof(x) == "language"
  is.language()symbolexpressionis.call()
  is.call(cl)class(cl)"call"cliforwhile({<-=class(cl)
as.call(x) "list""call"
  as.call()str2lang()parse(text=)call()as.call()parse()

as.call

```

```

call.Internal

```

```

do.callRecallis.languageexpressionfunction
callstr2langparse

```

```

is.call(call) #-> FALSE: Functions are NOT calls

## set up a function call to round with argument 10.5
cl <- call("round", 10.5)
is.call(cl) # TRUE
cl
identical(quote(round(10.5)), # <- less functional, but the same
  cl) # TRUE
## such a call can also be evaluated.
eval(cl) # [1] 10

class(cl) # "call"
typeof(cl)# "language"
is.call(cl) && is.language(cl) # always TRUE for "call"s

A <- 10.5
call("round", A)          # round(10.5)
call("round", quote(A)) # round(A)
f <- "round"
call(f, quote(A))        # round(A)
## if we want to supply a function we need to use as.call or similar
f <- round
## Not run: call(f, quote(A)) # error: first arg must be character
(g <- as.call(list(f, quote(A))))
eval(g)
## alternatively but less transparently
g <- list(f, quote(A))
mode(g) <- "call"

```

```

g
eval(g)

## Special calls (and some regular ones):
L <- as.list(E <- setNames( , c("if", "for", "while", "repeat", "function",
                              "(", "{", "[", "<=", "<<=", "->", "=)))
for(i in seq_along(L)) L[[i]] <- call(E[[i]]) # instead of lapply(E, call) ..
list_ <- function (...) `names<-`(list(...), vapply(sys.call()[-1L], as.character, ""))
(Tab <- noquote(sapply(list_(is.call, typeof, class, mode), \(F) sapply(L, F))))
## The 7 exceptions:
Tab[ Tab[, "class"] != "call" , c(3:4, 1:2)]

## see also the examples in the help for do.call

```

callCC

```
callCC(fun)
```

```
fun
```

```
callCCcallCCfunfunccallCCcallCC
```

```

# The following all return the value 1
callCC(function(k) 1)
callCC(function(k) k(1))
callCC(function(k) {k(1); 2})
callCC(function(k) repeat k(1))

```

CallExternal

```

.Call(.NAME, ..., PACKAGE)
.External(.NAME, ..., PACKAGE)

```

```
.NAME      "NativeSymbolInfo""RegisteredNativeSymbol""NativeSymbol"
...      .Call
PACKAGE    .NAME.so.dll
...      ...
```

```
.External
.NAME....NAME
```

```
Rinternals.hRdefines.h
```

```
PACKAGE.NAME
PACKAGE = "base"
PACKAGE = ""
```

```
.Call
```

```
dyn.load.C.Fortran
```

capabilities

```
capabilities(what = NULL,
             Xchk = any(nas %in% c("X11", "jpeg", "png", "tiff")))
```

```
what      NULLNULL
Xchk      logicalNA
```



```

jpeg          jpeg
png           png
tiff          tiff
tcltk         "X11"
X11           X11X11
aqua          quartz
              .Platform$GUI == "AQUA"R.app

http/ftp      urldownload.filehttp://ftp://TRUE"libcurl"ftp://
sockets       make.socketTRUE
libxml        libxmlTRUEFALSE
fifo
cledit        readline--no-readline--interactive
iconv         iconv
NLS
Rprof         Rprof()--enable-R-profiling
profmem       tracemem
cairo         svgcairo_pdfcairo_pstype = "cairo"bmpjpegpngtiffX11X11
ICU           icuSetCollate
long.double   Clong doubledouble--disable-long-double
              .Machine
libcurl       libcurlcurlGetHeadersdownload.fileurl

"jpeg""png""tiff"capabilities("aqua")type = "quartz"tiffcapabilities("aqua") ||
capabilities("tiff")

.PlatformextSoftVersiongrSoftVersion

capabilities()

if(!capabilities("ICU"))
  warning("ICU is not available")

## Does not call the internal X11-checking function:
capabilities(Xchk = FALSE)

## See also the examples for 'connections'.

```

cat

catprint

```
cat(... , file = "", sep = " ", fill = FALSE, labels = NULL,
     append = FALSE)
```

```
...
file      ""catsink"|cmd"cmd
sep
fill      FALSE"\n"widthfillTRUEfillfillfill
labels    fillFALSE
append    file"|cmd"TRUEfilefile
```

```
catsep =
"\n"fillTRUE
file"wt"
NULLprint.defaultencodeStringcatas.characterformatcat
catprintas.characteroptions"digits""scipen"
```

NULL

sep

printformatpaste

```
iter <- stats::rpois(1, lambda = 10)
## print an informative message
cat("iteration = ", iter <- iter + 1, "\n")

## 'fill' and label lines:
cat(paste(letters, 100* 1:26), fill = TRUE, labels = paste0("{", 1:10, "}:"))
```

`cbind`

```
cbind(..., deparse.level = 1)
rbind(..., deparse.level = 1)
## S3 method for class 'data.frame'
rbind(..., deparse.level = 1, make.row.names = TRUE,
      stringsAsFactors = FALSE, factor.exclude = TRUE)

...          "data.frame"cbinddata.framestringsAsFactors
deparse.level
      deparse.level = 0
      deparse.level = 1deparse.level = 2
make.row.names row.names
stringsAsFactors
      as.data.frame...data.framecharacter
factor.exclude TRUENAfactor.exclude = NA

cbindrbindisS4(.)cbindrbindcbind2rbind2deparse.level
vector
warning

cbindrbindNULL
231

...NULLNULL

cbindrbindcolnamesrownamesdeparse.level > 0deparse.level = 1is.symbol
cbind
rbind

cbinddata.frame(..., check.names = FALSE)
rbind
jfactor(., exclude = X(j))

  X(j) := if(isTRUE(factor.exclude)) {
    if(!NA.lev[j]) NA # else NULL
  } else factor.exclude

NA.lev[j]factorjNA
```

UseMethod()rbind.default

.../src/main/bind.c

cdata.frame

```
m <- cbind(1, 1:7) # the '1' (= shorter vector) is recycled
m
m <- cbind(m, 8:14)[, c(1, 3, 2)] # insert a column
m
cbind(1:7, diag(3)) # vector is subset -> warning

cbind(0, rbind(1, 1:3))
cbind(I = 0, X = rbind(a = 1, b = 1:3)) # use some names
xx <- data.frame(I = rep(0,2))
cbind(xx, X = rbind(a = 1, b = 1:3)) # named differently

cbind(0, matrix(1, nrow = 0, ncol = 4)) #> Warning (making sense)
dim(cbind(0, matrix(1, nrow = 2, ncol = 0))) #> 2 x 1

## deparse.level
dd <- 10
rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 0) # middle 2 rownames
rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 1) # 3 rownames (default)
rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 2) # 4 rownames

## cheap row names:
b0 <- gl(3,4, labels=letters[1:3])
bf <- setNames(b0, paste0("o", seq_along(b0)))
df <- data.frame(a = 1, B = b0, f = gl(4,3))
df. <- data.frame(a = 1, B = bf, f = gl(4,3))
new <- data.frame(a = 8, B = "B", f = "1")
(df1 <- rbind(df, new))
(df.1 <- rbind(df., new))
stopifnot(identical(df1, rbind(df, new, make.row.names=FALSE)),
           identical(df.1, rbind(df., new, make.row.names=FALSE)))
```

`char.expand`

```
char.expand(input, target, nomatch = stop("no match"))
```

`input`

`target`

`nomatch`

`targetnomatch`

[charmatchpmatch](#)

```
locPars <- c("mean", "median", "mode")
char.expand("me", locPars, warning("Could not expand!"))
char.expand("mo", locPars)
```

`character`

`"character"`

```
character(length = 0)
as.character(x, ...)
is.character(x)
```

`length`

`x`

`...`

```
as.character
is.character
as.character
as.vector
as.character
DBL_DIGformat
```

```
character""
as.characteras.vectorAbc
is.characterTRUEFALSE
```

```
as.character
```

```
optionscscipenOutDec
pastesubstrstrsplitchartrsubgrephelp.search(keyword = "character")
deparseas.character
Quotescharacter
```

```
form <- y ~ a + b + c
as.character(form) ## length 3
deparse(form)      ## like the input

a0 <- 11/999          # has a repeating decimal representation
(a1 <- as.character(a0))
format(a0, digits = 16) # shows 1 to 2 more digit(s)
a2 <- as.numeric(a1)
a2 - a0                # normally around -1e-17
as.character(a2)        # possibly different from a1
print(c(a0, a2), digits = 16)

as.character(list(A = "Abc", xy = c("x", "y"))) # "Abc" "c(\"x\", \"y\")"
## i.e., "Abc" directly instead of deparsing to "\"Abc\""
```

```
charmatch
```

```
charmatch
```

```
charmatch(x, table, nomatch = NA_integer_)
```

```
x                as.character
table
nomatch
```

`0nomatch`

`NA"NA"`

`xtablenomatch`

[pmatchmatch](#)

[startsWithgrepregexpr](#)

```
charmatch("", "") # returns 1
charmatch("m", c("mean", "median", "mode")) # returns 0
charmatch("med", c("mean", "median", "mode")) # returns 2
```

`chartr`

```
chartr(old, new, x)
tolower(x)
toupper(x)
casefold(x, upper = FALSE)
```

`x` [as.character](#)

`old`

`new`

`upper`

`chartrxoldnewoldnew`

`tolowertoupper`

`casefoldtolowertoupper`

`x`

[Encoding](#)

subgsub

```
x <- "MiXeD cAsE 123"
chartr("iXs", "why", x)
chartr("a-cX", "D-Fw", x)
tolower(x)
toupper(x)

## "Mixed Case" Capitalizing - toupper( every first letter of a word ) :

.simpleCap <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substring(s, 1, 1)), substring(s, 2),
        sep = "", collapse = " ")
}
.simpleCap("the quick red fox jumps over the lazy brown dog")
## -> [1] "The Quick Red Fox Jumps Over The Lazy Brown Dog"

## and the better, more sophisticated version:
capwords <- function(s, strict = FALSE) {
  cap <- function(s) paste(toupper(substring(s, 1, 1)),
                           {s <- substring(s, 2); if(strict) tolower(s) else s},
                           sep = "", collapse = " ")
  sapply(strsplit(s, split = " "), cap, USE.NAMES = !is.null(names(s)))
}
capwords(c("using AIC for model selection"))
## -> [1] "Using AIC For Model Selection"
capwords(c("using AIC", "for MODEL selection"), strict = TRUE)
## -> [1] "Using Aic" "For Model Selection"
##          ^^^          ^^^^^
##          'bad'        'good'

## -- Very simple insecure crypto --
rot <- function(ch, k = 13) {
  p0 <- function(...) paste(c(...), collapse = "")
  A <- c(letters, LETTERS, " ")
  I <- seq_len(k); chartr(p0(A), p0(c(A[-I], A[I])), ch)
}

pw <- "my secret pass phrase"
(crypw <- rot(pw, 13)) #-> you can send this off

## now ``decrypt`` :
rot(crypw, 54 - 13) # -> the original:
stopifnot(identical(pw, rot(crypw, 54 - 13)))
```

chkDots

```
.....chkDots()
```

```
chkDots(..., which.call = -1, allowed = character(0))
```

```
...
```

```
which.call      sys.call()
```

```
allowed        ...
```

```
warning...
```

```
seq.default ## <- you will see ' chkDots(...) '
```

```
seq(1,5, foo = "bar") # gives warning via chkDots()
```

```
## warning with more than one ...-entry:
```

```
density.f <- function(x, ...) NextMethod("density")
```

```
x <- density(structure(rnorm(10), class="f"), bar=TRUE, baz=TRUE)
```

chol

```
chol(x, ...)
```

```
## Default S3 method:
```

```
chol(x, pivot = FALSE, LINPACK = FALSE, tol = -1, ...)
```

```
x
```

```
...
```

```
pivot
```

```
LINPACK
```

```
tol           pivot = TRUE
```

```
chol

$$xR'R = xx$$

pivot = FALSExx
pivot = TRUExxattr(Q, "rank")attr(Q, "pivot")t(Q) %*% Qxpivot <- attr(Q, "pivot")
oo <- order(pivot)t(Q[, oo]) %*% Q[, oo]xt(Q) %*% Qx[pivot, pivot]
tolnrow(x) * .Machine$double.neg.eps * max(diag(x))tol
```

```

$$RR'R = x$$

"pivot""rank"
```

```
pivot = TRUExpivot = TRUEx
```

```
DPOTRFDPSTRF
https://netlib.org/lapack/
```

```
https://netlib.org/lapack/lug/lapack\_lug.html
```

```
chol2invbacksolve
qrsvd
```

```
( m <- matrix(c(5,1,1,3),2,2) )
( cm <- chol(m) )
t(cm) %*% cm #-- = 'm'
crossprod(cm) #-- = 'm'

# now for something positive semi-definite
x <- matrix(c(1:5, (1:5)^2), 5, 2)
x <- cbind(x, x[, 1] + 3*x[, 2])
colnames(x) <- letters[20:22]
m <- crossprod(x)
qr(m)$rank # is 2, as it should be

# chol() may fail, depending on numerical rounding:
# chol() unlike qr() does not use a tolerance.
try(chol(m))

(Q <- chol(m, pivot = TRUE))
## we can use this by
```

```

pivot <- attr(Q, "pivot")
crossprod(Q[, order(pivot)]) # recover m

## now for a non-positive-definite matrix
( m <- matrix(c(5,-5,-5,3), 2, 2) )
try(chol(m)) # fails
(Q <- chol(m, pivot = TRUE)) # warning
crossprod(Q) # not equal to m

```

chol2inv

$$(X'X)^{-1}RX$$

```
chol2inv(x, size = NCOL(x), LINPACK = FALSE)
```

```

x                size
size            x
LINPACK

```

DPOTRI<https://netlib.org/lapack/>

https://netlib.org/lapack/lug/lapack_lug.html

cholsolve

```

cma <- chol(ma <- cbind(1, 1:3, c(1,3,7)))
ma %*% chol2inv(cma)

```

chooseOpsMethod

chooseOpsMethodOps

reverse = FALSExychooseOpsMethod()FALSEExchooseOpsMethodxymxmyreverse = TRUE

chooseOpsMethod(x, y, mx, my, cl, reverse)

xy

mxmy xy

cl

reverse xy

TRUEFALSETRUEmx

Ops

```
# Create two objects with custom Ops methods
```

```
foo_obj <- structure(1, class = "foo")
```

```
bar_obj <- structure(1, class = "bar")
```

```
`+.foo` <- function(e1, e2) "foo"
```

```
Ops.bar <- function(e1, e2) "bar"
```

```
invisible(foo_obj + bar_obj) # Warning: Incompatible methods
```

```
chooseOpsMethod.bar <- function(x, y, mx, my, cl, reverse) TRUE
```

```
stopifnot(exprs = {  
  identical(foo_obj + bar_obj, "bar")  
  identical(bar_obj + foo_obj, "bar")  
})
```

```
# cleanup
```

```
rm(foo_obj, bar_obj, `+.foo`, Ops.bar, chooseOpsMethod.bar)
```

class

```
class(x)
class(x) <- value
unclass(x)
inherits(x, what, which = FALSE)
nameOfClass(x)
isa(x, what)
```

```
oldClass(x)
oldClass(x) <- value
.class2(x)
```

```
x
whatvalue      valueNULLwhatnameOfClass()
which
```

```
classoldClassoldClass<-
"matrix""array""function""numeric"typeof(x)mode(x)"language"mode"call"calliffor
while({<-=
xfoo(x)class(x)foo.numeric()characterUseMethod().class2(x)
.class2()
NULL"NULL"class
func("first", "second")fun.firstfun.secondfun.default
classclass<-NULLoldClass<-NULLclass(x) <- setdiff(class(x), "ts")
unclass
inheritswhatwhichTRUEwhatclass(x)whatwhichFALSETRUEinheritswhatclass
nameOfClassinheritswhatwhatnameOfClass
isaxwhatisxTRUEclass(x)what
inheritsisa
```

```
classinherits
as(object, value)
inheritsisisinherits
```

```
UseMethodclassoldClassoldClassis.object
```

UseMethodNextMethod

```
x <- 10
class(x) # "numeric"
oldClass(x) # NULL
inherits(x, "a") #FALSE
class(x) <- c("a", "b")
inherits(x,"a") #TRUE
inherits(x, "a", TRUE) # 1
inherits(x, c("a", "b", "c"), TRUE) # 1 2 0

class( quote(pi) )          # "name"
## regular calls
class( quote(sin(pi*x)) )   # "call"
## special calls
class( quote(x <- 1) )      # "<-"
class( quote((1 < 2)) )     # "("
class( quote( if(8<3) pi ) ) # "if"

.class2(pi)                 # "double" "numeric"
.class2(matrix(1:6, 2,3)) # "matrix" "array" "integer" "numeric"
```

col

```
col(x, as.factor = FALSE)
.col(dim)
```

```
x          dim
dim
as.factor
```

```
xijjj
```

rowslice.index

```

# extract an off-diagonal of a matrix
ma <- matrix(1:12, 3, 4)
ma[row(ma) == col(ma) + 1]

# create an identity 5-by-5 matrix more slowly than diag(n = 5):
x <- matrix(0, nrow = 5, ncol = 5)
x[row(x) == col(x)] <- 1

(i34 <- .col(3:4))
stopifnot(identical(i34, .col(c(3,4)))) # 'dim' maybe "double"

```

Colon

```

from:to
a:b

```

```

from
to
ab          factor

```

```

:a:binteraction(a, b)
from:toseq(from, to)fromto1-1tofrom1e-7

```

```

integerfrom"double"mode"numeric"
la:lblb

```

```

:
```

```

seqfrom:to
:interaction
:formula

```

```

1:4
pi:6 # real
6:pi # integer

```

```

f1 <- gl(2, 3); f1
f2 <- gl(3, 2); f2
f1:f2 # a factor, the "cross" f1 x f2

```

colSums

```
colSums(x, na.rm = FALSE, dims = 1)
rowSums(x, na.rm = FALSE, dims = 1)
colMeans(x, na.rm = FALSE, dims = 1)
rowMeans(x, na.rm = FALSE, dims = 1)
```

```
.colSums(x, m, n, na.rm = FALSE)
.rowSums(x, m, n, na.rm = FALSE)
.colMeans(x, m, n, na.rm = FALSE)
.rowMeans(x, m, n, na.rm = FALSE)
```

```
x          .colSums()m * n
na.rm      NaN
dims       row*dims+1, ...col*1:dims
mn         x.colSums()
```

```
applyFUN = meanFUN = sumNaNNa.rm = FALSENaNNaNaNA
na.omitcomplete.casesx
.colSums()
```

```
dimnamesnames
na.rm = TRUE0*SumsNaN*Meanssummean
```

[applyrowsum](#)

```
## Compute row and column sums for a matrix:
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
rowSums(x); colSums(x)
dimnames(x)[[1]] <- letters[1:8]
rowSums(x); colSums(x); rowMeans(x); colMeans(x)
x[] <- as.integer(x)
rowSums(x); colSums(x)
x[] <- x < 3
rowSums(x); colSums(x)
x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
x[3, ] <- NA; x[4, 2] <- NA
rowSums(x); colSums(x); rowMeans(x); colMeans(x)
rowSums(x, na.rm = TRUE); colSums(x, na.rm = TRUE)
```



```

rowMeans(x, na.rm = TRUE); colMeans(x, na.rm = TRUE)

## an array
dim(UCBAdmissions)
rowSums(UCBAdmissions); rowSums(UCBAdmissions, dims = 2)
colSums(UCBAdmissions); colSums(UCBAdmissions, dims = 2)

## complex case
x <- cbind(x1 = 3 + 2i, x2 = c(4:1, 2:5) - 5i)
x[3, ] <- NA; x[4, 2] <- NA
rowSums(x); colSums(x); rowMeans(x); colMeans(x)
rowSums(x, na.rm = TRUE); colSums(x, na.rm = TRUE)
rowMeans(x, na.rm = TRUE); colMeans(x, na.rm = TRUE)

```

commandArgs

```
commandArgs(trailingOnly = FALSE)
```

```
trailingOnly  --args
```

```
--args
```

```
trailingOnly = TRUE--args
```

[R.home\(\)](#)[StartupBATCH](#)

```

commandArgs()
## Spawn a copy of this application as it was invoked,
## subject to shell quoting issues
## system(paste(commandArgs(), collapse = " "))

```

comment	"comment"
---------	-----------

```
data.frame
attributescommentprintprint.default
NULL
```

```
comment(x)
comment(x) <- value
```

```
x
value          characterNULL
```

```
attributesattr
```

```
x <- matrix(1:12, 3, 4)
comment(x) <- c("This is my very important data from experiment #0234",
               "Jun 5, 1998")
x
comment(x)
```

Comparison

```
x < y
x > y
x <= y
x >= y
x == y
x != y
```

```
xy
```

```
OpsOps
localesen_USCZSTaazngg
Encoding
```

```
xy
```

```
NANaNA
==!=identical
```

`CompareOp``sc(e1, e2)`

`==``!``if``TRUE``FALSE``identical`

`==``!``all.equal``identical``is``TRUE`

``<``(`x`, `y`)`Ops`

https://en.wikipedia.org/wiki/Collating_sequence<https://unicode.org/reports/tr10/><https://icu.unicode.org/>

`Logic`

`factor`

`Syntax`

`capabilities``icuSetCollate`

```
x <- stats::rnorm(20)
```

```
x < 1
```

```
x[x > 0]
```

```
x1 <- 0.5 - 0.3
```

```
x2 <- 0.3 - 0.1
```

```
x1 == x2 # FALSE on most machines
```

```
isTRUE(all.equal(x1, x2)) # TRUE everywhere
```

```
# range of most 8-bit charsets, as well as of Latin-1 in Unicode
```

```
z <- c(32:126, 160:255)
```

```
x <- if(l10n_info()$MBCS) {
```

```
  intToUtf8(z, multiple = TRUE)
```

```
} else rawToChar(as.raw(z), multiple = TRUE)
```

```
## by number
```

```
writelines(strwrap(paste(x, collapse=" "), width = 60))
```

```
## by locale collation
```

```
writelines(strwrap(paste(sort(x), collapse=" "), width = 60))
```

complex

 $+-*/^$

```
complex(length.out = 0, real = numeric(), imaginary = numeric(),
        modulus = 1, argument = 0)
```

```
as.complex(x, ...)
```

```
is.complex(x)
```

 $\operatorname{Re}(z)$ $\text{Im}(z)$
$$\text{Mod}(z)$$
 $\text{Arg}(z)$ $\text{Conj}(z)$

length.out

real

imaginary

modulus

argument

x	complex
---	---------

$$z \quad \text{complex}$$

...

complex

```
as.complexas.vectoras.complex(x)x"logical""integer""double"NANANaNNANA_complex_
```

```
NANANANaNNaNNANANaNis.na(.)NA_complex_
```

```
is.complexis.numericTRUE
```

$$\text{ReImModArgConj}z = x + iyxyr = \text{Mod}(z) = \sqrt{x^2 + y^2}\phi = \text{Arg}(z)x = r \cos(\phi)y = r \sin(\phi)$$

Complex

 $+-*/^$

```
%%crossprodtcrossprodmatrixsolveeigensvd
```

NaNANA_complex_

as.complex

ReImModArgConjComplex

```
NaNdouble complexcomplex(re=NaN, im=NaN)+-
```

NANA_complex_NA_complex_

```
as.complex("1i")1iNA_complex_
```

Arithmeticpolyrootnn

```
require(graphics)

0i ^ (-3:3)

matrix(1i^ (-6:5), nrow = 4) #- all columns are the same
0 ^ 1i # a complex NaN

## create a complex normal vector
z <- complex(real = stats::rnorm(100), imaginary = stats::rnorm(100))
## or also (less efficiently):
z2 <- 1:2 + 1i*(8:9)

## The Arg(.) is an angle:
zz <- (rep(1:4, length.out = 9) + 1i*(9:1))/10
zz.shift <- complex(modulus = Mod(zz), argument = Arg(zz) + pi)
plot(zz, xlim = c(-1,1), ylim = c(-1,1), col = "red", asp = 1,
     main = expression(paste("Rotation by ", " ", pi == 180^o)))
abline(h = 0, v = 0, col = "blue", lty = 3)
points(zz.shift, col = "orange")

## as.complex(<some NA>): numbers keep Im = 0:
NAs <- vapply(list(NA, NA_integer_, NA_real_, NA_character_, NA_complex_),
              as.complex, 0+0i)
stopifnot(is.na(NAs), is.na(Re(NAs))) # has always been true
showC <- function(z) noquote(paste0("(", Re(z), ", ", Im(z), ")"))
showC(NAs)
Im(NAs) # [0 0 0 NA NA] \\ in R <= 4.3.x was [NA NA 0 NA NA]
stopifnot(Im(NAs)[1:3] == 0)
stopifnot(Im(NA_real_ + 0i) == 0) # has always been true

## The exact result of this *depends* on the platform, compiler, math-library:
(NpNA <- NaN + NA_complex_); str(NpNA) # *behaves* as 'cplx NA' ..
stopifnot(is.na(NpNA), is.na(NA_complex_), is.na(Re(NA_complex_)), is.na(Im(NA_complex_)))
showC(NpNA) # but does not always show '(NaN,NA)'
## and this is not TRUE everywhere:
identical(NpNA, NA_complex_)
showC(NA_complex_) # always == (NA,NA)
```

conditions

```

tryCatch(expr, ..., finally)
withCallingHandlers(expr, ...)
globalCallingHandlers(...)

signalCondition(cond)

simpleCondition(message, call = NULL)
simpleError      (message, call = NULL)
simpleWarning    (message, call = NULL)
simpleMessage    (message, call = NULL)

errorCondition(message, ..., class = NULL, call = NULL)
warningCondition(message, ..., class = NULL, call = NULL)

## S3 method for class 'condition'
as.character(x, ...)
## S3 method for class 'error'
as.character(x, ...)
## S3 method for class 'condition'
print(x, ...)
## S3 method for class 'restart'
print(x, ...)

conditionCall(c)
## S3 method for class 'condition'
conditionCall(c)
conditionMessage(c)
## S3 method for class 'condition'
conditionMessage(c)

withRestarts(expr, ...)

computeRestarts(cond = NULL)
findRestart(name, cond = NULL)
invokeRestart(r, ...)
tryInvokeRestart(r, ...)
invokeRestartInteractively(r)

isRestart(x)
restartDescription(r)
restartFormals(r)

suspendInterrupts(expr)
allowInterrupts(expr)

.signalSimpleWarning(msg, call)
.handleSimpleError(h, msg, call)
.tryResumeInterrupt()

```

call
cond
expr
finally
h
message
msg
name
r
x
class
...

conditionerrorwarningsimpleErrorstopsimpleWarningwarningsimpleMessagemessage
conditionMessageconditionCall
errorCondition...warningCondition
signalConditionstopwarning
tryCatch...finallytryCatchtryCatchfinally
...tryCatchexprtryCatch
exprtryCatchtryCatchtryCatch
withCallingHandlerssignalConditionsignalConditionNULL
globalCallingHandlerswithCallingHandlerserrorretry_loadNamespace
withCallingHandlerstryCatchglobalCallingHandlersoptionsglobalCallingHandlers
interruptcondition
withRestartsabort
findRestartcomputeRestartsfindRestartcomputeRestarts
invokeRestartinvokeRestartinvokeRestartfindRestart
tryInvokeRestartinvokeRestartfindRestart"muffleWarning"stopmessageinvokeRestart
withRestartsname = functionname = stringdescriptionfindRestartNULLhandler
descriptiontesttestTRUEFALSETRUE
interactiveinvokeRestartInteractivelyinteractive
suspendInterruptsallowInterrupts
.signalSimpleWarning.handleSimpleError.tryResumeInterrupt

tryCatch

[stopwarningtrytryCatchassertCondition](#)

```

tryCatch(1, finally = print("Hello"))
e <- simpleError("test error")
## Not run:
  stop(e)
  tryCatch(stop(e), finally = print("Hello"))
  tryCatch(stop("fred"), finally = print("Hello"))

## End(Not run)
tryCatch(stop(e), error = function(e) e, finally = print("Hello"))
tryCatch(stop("fred"), error = function(e) e, finally = print("Hello"))
withCallingHandlers({ warning("A"); 1+2 }, warning = function(w) {})
## Not run:
  { withRestarts(stop("A"), abort = function() {}); 1 }

## End(Not run)
withRestarts(invokerestart("foo", 1, 2), foo = function(x, y) {x + y})

##--> More examples are part of
##--> demo(error.catching)

```

conflicts

conflicts[search](#)

```
conflicts(where = search(), detail = FALSE)
```

where

```
detail          TRUE
```

```
detail = FALSEdetail = TRUE
```

```

lm <- 1:3
conflicts(, TRUE)
## gives something like
# $.GlobalEnv
# [1] "lm"
#
# $package:base
# [1] "lm"

```

```

## Remove things from your "workspace" that mask others:
remove(list = conflicts(detail = TRUE)$GlobalEnv)

```

connections

```
file(description = "", open = "", blocking = TRUE,
      encoding = getOption("encoding"), raw = FALSE,
      method = getOption("url.method", "default"))

url(description, open = "", blocking = TRUE,
     encoding = getOption("encoding"),
     method = getOption("url.method", "default"),
     headers = NULL)

gzfile(description, open = "", encoding = getOption("encoding"),
        compression = 6)

bzfile(description, open = "", encoding = getOption("encoding"),
        compression = 9)

xzfile(description, open = "", encoding = getOption("encoding"),
        compression = 6)

zstdfile(description, open = "", encoding = getOption("encoding"),
          compression = 9)

unz(description, filename, open = "", encoding = getOption("encoding"))

pipe(description, open = "", encoding = getOption("encoding"))

fifo(description, open = "", blocking = FALSE,
      encoding = getOption("encoding"))

socketConnection(host = "localhost", port, server = FALSE,
                 blocking = FALSE, open = "a+",
                 encoding = getOption("encoding"),
                 timeout = getOption("timeout"),
                 options = getOption("socketOptions"))

serverSocket(port)

socketAccept(socket, blocking = FALSE, open = "a+",
             encoding = getOption("encoding"),
             timeout = getOption("timeout"),
             options = getOption("socketOptions"))

open(con, ...)
## S3 method for class 'connection'
open(con, open = "r", blocking = TRUE, ...)
```

```
close(con, ...)
## S3 method for class 'connection'
close(con, type = "rw", ...)
```

```
flush(con)
```

```
isOpen(con, rw = "")
isIncomplete(con)
```

```
socketTimeout(socket, timeout = -1)
```

```
description
```

```
open
```

```
blocking
```

```
encoding
```

```
raw
```

```
method          c("default", "internal", "wininet", "libcurl")
```

```
headers          User-AgentHTTPUserAgentoptions
```

```
compression      zstdfilexzfilezstdfile
```

```
timeout
```

```
options          "no-delay"
```

```
filename
```

```
host
```

```
port
```

```
server
```

```
socket
```

```
con
```

```
type
```

```
rw              "read" "write"
```

```
...
```

```
socketConnectionsocketAcceptserverSocketopen
```

```
fileurl"" "clipboard" "stdin" RGuistdin()stdinnullfile()
```

```
urlhttp://https://ftp://file://"internal""wininet""libcurl"https://curl.se/libcurl/"default""internal"file://"libcurl""internal"file://"wininet"file://  
http://https://download.file
```

```
gzfilegzipbzip2xzlmazstd
```

```
bzfilebzip2
```

```
xzfilexzhttps://en.wikipedia.org/wiki/Xz\_lzmahttps://en.wikipedia.org/wiki/LZMA
```

```
zstdfilezstdhttps://en.wikipedia.org/wiki/Zstd
```

```
unz.zip
```

pipeCOMSPEC
fifofifo
filegzfilehttp://https://
opencloseseek
openopen
close
flushstdoutstderr
filefifo""w+open = "w+b"
socketConnection(server=TRUE)serverSocketsocketAcceptclose
socketConnectionsocketAccept"no-delay"TCP_NODELAY
socketTimeouttimeout

filepipefifourlgzfilebzfilexzfilezstdfileunzsocketConnectionsocketAccept
serverSocket"connection"
openflushNULL
closeNULLon.exit()
isOpen
isIncompletereadLines
socketTimeout

urlfilefile://http://https://ftp://
method = "libcurl"libcurlVersionhttps://https://
ftp://"internal"
http://URLencode"wininet"
file://methodfile://host/path/to/filehost
file:///path/to/filepath/to/file//path/to/file
file:///d:/R/reposfile://path_to_file
file:URLdecode

download.file

open
"r""rt"
"w""wt"
"a""at"
"rb"
"wb"
"ab"

"r+" "r+b"

"w+" "w+b"

"a+" "a+b"

umaskSys.umask

readLinessourcepushBackreadBinloadsave

open = ""open"r"gzfilebzfilexzfilezstdfile"rb"

gzfilebzfilexzfilezstdfile

gzipbzip2zstdxzlzma

file(raw = FALSE)open""r""rt""rb"gzfile

openopen = "w"unlink

compressxzfile(compress = 9)xzfilecompressxz-e6xz

zstd--fast=

gzipbzip2xbzip2gzipxzsave.rdaxzcompress = 9

zstdxz

iconv""native.enc"

writeLineswriteLinesuseBytes=TRUEencoding""native.enc"

encodingreadLinesscanencodingreadLinesscanEncoding"UTF-8""native.enc"encoding
"UTF-8""latin1"encodingreadLinesscan

readLinesreadChar

"UCS-2LE""UTF-16LE"0xFEFFiconv"UCS-2"glibc"UTF-16""UCS-2"<https://en.wikipedia.org/wiki/UTF-16>"UCS-2LE"

"UTF-8-BOM"writeChar("\uffeff", con, eos = NULL)writeBin(as.raw(c(0xef, 0xbb, 0xbf)), binary_con)

"utf8""mac""macroman""UTF-8""macintosh""utf8""UTF-8"

Sys.getlocale("LC_CTYPE")

readLines

options("timeout")http:ftp:

file()

```
filedescription      =      "clipboard""r"https://specifications.freedesktop.org/clipboard-spec/latest"X11_primary""X11_secondary""X11_clipboard"
```

```
xcliphttps://github.com/astrand/xclipxselhttps://www.vergenet.net/~conrad/software/xsel/pipe("xclip -i", "w")  
pipe("pbpaste")pipe("pbcopy", "w")
```

```
filepipedescription
```

```
ftp://
```

```
https://
```

```
libcurllibcurllibcurlVersion()
```

```
stdinstdoutstderr--max-connections=N
```

```
ulimit -ndyn.load
```

```
filegzfilebzfilexzfilezstdfilepipeurlunz
```

```
makeCluster
```

```
"r""r+""x"
```

```
vsnprintfifogzfilebzfilexzfilezstdfile
```

```
https://www.r-project.org/doc/Rnews/Rnews\_2001-1.pdf
```

```
textConnectionseekshowConnectionspushBack
```

```
readLineswriteLinescatsinkscanparseread.dcfdumpreadBinreadCharwriteBin  
writeCharloadsave
```

```
capabilitiesfifo
```

```
gzcongzip
```

```
optionsHTTPUserAgentinternet.infotimeout
```

```
memCompress
```

```
extSoftVersionzlibgzfilebzzip2xz
```

```
flush.console
```

```

zzfil <- tempfile(fileext=".data")
zz <- file(zzfil, "w") # open an output file connection
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = zz, sep = "\n")
cat("One more line\n", file = zz)
close(zz)
readLines(zzfil)
unlink(zzfil)

zzfil <- tempfile(fileext=".gz")
zz <- gzfile(zzfil, "w") # compressed file
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = zz, sep = "\n")
close(zz)
readLines(zz <- gzfile(zzfil))
close(zz)
unlink(zzfil)
zz # an invalid connection

zzfil <- tempfile(fileext=".bz2")
zz <- bzfile(zzfil, "w") # bzip2-ed file
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = zz, sep = "\n")
close(zz)
zz # print() method: invalid connection
print(readLines(zz <- bzfile(zzfil)))
close(zz)
unlink(zzfil)

## An example of a file open for reading and writing
Tpath <- tempfile("test")
Tfile <- file(Tpath, "w+")
c(isOpen(Tfile, "r"), isOpen(Tfile, "w")) # both TRUE
cat("abc\ndef\n", file = Tfile)
readLines(Tfile)
seek(Tfile, 0, rw = "r") # reset to beginning
readLines(Tfile)
cat("ghi\n", file = Tfile)
readLines(Tfile)

Tfile # -> print() : "valid" connection
close(Tfile)
Tfile # -> print() : "invalid" connection
unlink(Tpath)

## We can do the same thing with an anonymous file.
Tfile <- file()
cat("abc\ndef\n", file = Tfile)
readLines(Tfile)
close(Tfile)

## Not run: ## fifo example -- may hang even with OS support for fifos
if(capabilities("fifo")) {
  zzfil <- tempfile(fileext="-fifo")
  zz <- fifo(zzfil, "w+")
  writeLines("abc", zz)
  print(readLines(zz))
  close(zz)
  unlink(zzfil)
}

```

```

}
## End(Not run)

## Unix examples of use of pipes

# read listing of current directory
readLines(pipe("ls -l"))

# remove trailing commas. Suppose

## Not run: % cat data2_
450, 390, 467, 654, 30, 542, 334, 432, 421,
357, 497, 493, 550, 549, 467, 575, 578, 342,
446, 547, 534, 495, 979, 479
## End(Not run)
# Then read this by
scan(pipe("sed -e s/,,$// data2_"), sep = ",")

# convert decimal point to comma in output: see also write.table
# both R strings and (probably) the shell need \ doubled
zzfil <- tempfile("outfile")
zz <- pipe(paste("sed s/\\\\\\. / >", zzfil), "w")
cat(format(round(stats::rnorm(48), 4)), fill = 70, file = zz)
close(zz)
file.show(zzfil, delete.file = TRUE)

## Not run:
## example for a machine running a finger daemon

con <- socketConnection(port = 79, blocking = TRUE)
writelines(paste0(system("whoami", intern = TRUE), "\r"), con)
gsub(" *$", "", readLines(con))
close(con)

## End(Not run)

## Not run:
## Two R processes communicating via non-blocking sockets
# R process 1
con1 <- socketConnection(port = 6011, server = TRUE)
writelines(LETTERS, con1)
close(con1)

# R process 2
con2 <- socketConnection(Sys.info()["nodename"], port = 6011)
# as non-blocking, may need to loop for input
readLines(con2)
while(isIncomplete(con2)) {
  Sys.sleep(1)
  z <- readLines(con2)
  if(length(z)) print(z)
}
close(con2)

## examples of use of encodings
# write a file in UTF-8

```

```
cat(x, file = (con <- file("foo", "w", encoding = "UTF-8"))); close(con)
# read a 'Windows Unicode' file
A <- read.table(con <- file("students", encoding = "UCS-2LE")); close(con)

## End(Not run)
```

Constants

```
LETTERS
letters
month.abb
month.name
pi
```

```
LETTERS
letters
month.abb
month.name
pi
```

[dataDateTimeClasses](#)

[QuotesNumericConstants](#)

```
## John Machin (ca 1706) computed pi to over 100 decimal places
## using the Taylor series expansion of the second term of
pi = 4*(4*atan(1/5) - atan(1/239))

## months in English
month.name
## months in your current locale
format(ISOdate(2000, 1:12, 1), "%B")
format(ISOdate(2000, 1:12, 1), "%b")
```

contributors

contributors()

Control

```
if(cond) expr
if(cond) cons.expr else alt.expr
```

```
for(var in seq) expr
while(cond) expr
repeat expr
break
next
```

```
x %||% y
```

```
cond          NA
var
seq           NULL
exprcons.expralt.exprxy
              { expr1 ; expr2 }
```

```
breakforwhilerepeatnextbreaknext
{ .. }if(..)for(...)}elseif ... elsesourceif
seqforseqvarseqvarvar
%||%x %||% y
```

```
if (is.null(x)) y else x
                                # or equivalently, of course,
if(!is.null(x)) x else y
```

```
ifNULLelse
forwhilerepeatNULLforvarseqNULL
breaknext
```

[SyntaxParen](#)
[ifelseswitch](#)

```
for(i in 1:5) print(1:i)
for(n in c(2,5,10,20,50)) {
  x <- stats::rnorm(n)
  cat(n, ": ", sum(x^2), "\n", sep = "")
}
f <- factor(sample(letters[1:5], 10, replace = TRUE))
for(i in unique(f)) print(i)

res <- {}
res %||% "alternative result"
x <- head(x) %||% stop("parsed, but *not* evaluated..")

res <- if(sum(x) > 7.5) mean(x) # may be NULL
res %||% "sum(x) <= 7.5"
```

copyright

[licensecontributors](#)

/COPYRIGHTS

crossprod

```
xyt(x) %*% ycrossprod %*% t(y)tcrossprod
matOps
```

```
crossprod(x, y = NULL, ...)
tcrossprod(x, y = NULL, ...)
```

```
xy          y = NULLx
...
```

```
dimnamesxy
```

```
xynamesdimnames
```

```
%*%
```

```
NaNInfoptions("matprod")
```

```
%*%%O%
```

```
(z <- crossprod(1:4))    # = sum(1 + 2^2 + 3^2 + 4^2)
drop(z)                  # scalar
x <- 1:4; names(x) <- letters[1:4]; x
tcrossprod(as.matrix(x)) # is
identical(tcrossprod(as.matrix(x)),
           crossprod(t(x)))
tcrossprod(x)            # no dimnames

m <- matrix(1:6, 2,3) ; v <- 1:3; v2 <- 2:1
stopifnot(identical(tcrossprod(v, m), v %*% t(m)),
           identical(tcrossprod(v, m), crossprod(v, t(m))),
           identical(crossprod(m, v2), t(m) %*% v2))
```

```
Cstack_info
```

```
Cstack_info()
```

```
sizeNA
```

```
glibc
```

```
options("expressions")
```

size NA
current NA
direction 1-1
eval_depth Cstack_info

Cstack_info()

cumsum

cumsum(x)
cumprod(x)
cummax(x)
cummin(x)

x cummincummax

Math

xcumprod*
NAxNAcumsumNAxNAcumsum()Im(cumsum(x))≡cumsum(Im(x))

cumsumcumprodMathcummaxcummin

cumsum

cumsum(1:10)
cumprod(1:10)
cummin(c(3:1, 2:0, 4:2))
cummax(c(3:1, 2:0, 4:2))

curlGetHeaders

```
http://ftp://https://https://
```

```
curlGetHeaders(url, redirect = TRUE, verify = TRUE,  
               timeout = 0L, TLS = "")
```

```
url
```

```
redirect
```

```
verify
```

```
timeout
```

```
TLS           https://""libcurl"1.1""1.2"libcurl"1.3"libcurl
```

```
curl -I -Lcurl -Iftp://
```

```
timeoutgetOption("timeout")
```

```
options(internet.info = 1)
```

```
status405
```

```
download.file
```

```
"status"
```

```
https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codeshttps://en.wikipedia.org/wiki/List\_of\_FTP\_server\_return\_codes
```

```
capabilities("libcurl")libcurlVersionlibcurl
```

```
optionsHTTPUserAgenttimeout
```

```
## needs Internet access, results vary
```

```
curlGetHeaders("http://bugs.r-project.org") ## this redirects to https://
```

```
## 2023-04: replaces slow and unreliable https://httpbin.org/status/404
```

```
curlGetHeaders("https://developer.R-project.org/inet-tests/not-found")
```

```
## returns status
```

cut

cutxx

cut(x, ...)

Default S3 method:

```
cut(x, breaks, labels = NULL,  
    include.lowest = FALSE, right = TRUE, dig.lab = 3,  
    ordered_result = FALSE, ...)
```

x

breaks x

labels "(a,b]"labels = FALSE

include.lowest right = FALSE

right

dig.lab

ordered_result

...

breaksbreaksx

```
labels"(b1, b2]"(b2, b3]"right = TRUE"[b1, b2]"right = FALSEdig.labb1b2"Range3"  
formatC
```

breakslabels

```
getOption("OutDec")labels = NULL
```

```
factorlabels = FALSE
```

breaksNANaNA

```
table(cut(x, br))hist(x, br, plot = FALSE)cut(*, labels = FALSE)findInterval()
```

[split](#)[factor](#)[tabulate](#)[table](#)[findInterval](#)

[quantile](#)

[.bincode](#)

```

Z <- stats::rnorm(10000)
table(cut(Z, breaks = -6:6))
sum(table(cut(Z, breaks = -6:6, labels = FALSE)))
sum(graphics::hist(Z, breaks = -6:6, plot = FALSE)$counts)

cut(rep(1,5), 4) #-- dummy
tx0 <- c(9, 4, 6, 5, 3, 10, 5, 3, 5)
x <- rep(0:8, tx0)
stopifnot(table(x) == tx0)

table( cut(x, breaks = 8))
table( cut(x, breaks = 3*(-2:5)))
table( cut(x, breaks = 3*(-2:5), right = FALSE))

##--- some values OUTSIDE the breaks :
table(cx <- cut(x, breaks = 2*(0:4)))
table(cxl <- cut(x, breaks = 2*(0:4), right = FALSE))
which(is.na(cx)); x[is.na(cx)] #-- the first 9 values 0
which(is.na(cxl)); x[is.na(cxl)] #-- the last 5 values 8

## Label construction:
y <- stats::rnorm(100)
table(cut(y, breaks = pi/3*(-3:3)))
table(cut(y, breaks = pi/3*(-3:3), dig.lab = 4))

table(cut(y, breaks = 1*(-3:3), dig.lab = 4))
# extra digits don't "harm" here
table(cut(y, breaks = 1*(-3:3), right = FALSE))
#- the same, since no exact INT!

## sometimes the default dig.lab is not enough to be avoid confusion:
aaa <- c(1,2,3,4,5,2,3,4,5,6,7)
cut(aaa, 3)
cut(aaa, 3, dig.lab = 4, ordered_result = TRUE)

## one way to extract the breakpoints
labs <- levels(cut(aaa, 3))
cbind(lower = as.numeric( sub("\\((.+),.*", "\\1", labs) ),
      upper = as.numeric( sub("[^,]*,([^\"]*)\\]", "\\1", labs) ))

```

cut.POSIXt

cut

```

## S3 method for class 'POSIXt'
cut(x, breaks, labels = NULL, start.on.monday = TRUE,
    right = FALSE, ...)

## S3 method for class 'Date'
cut(x, breaks, labels = NULL, start.on.monday = TRUE,
    right = FALSE, ...)

```

```

x          "POSIXt""Date"
breaks     x"sec""min""hour""day""DSTday""week""month""quarter""year""s"
          "Date""day""week""month""quarter""year"
labels     rightlabels = FALSE
start.on.monday
          breaks = "weeks"

right...

```

```

rightinclude.lowest = TRUE
breaks = "quarter"min(x)
breakslabels

```

```

labels = FALSE
breaksNANA

```

[seq.POSIXtseq.Datecut](#)

```

## random dates in a 10-week period
cut(ISOdate(2001, 1, 1) + 70*86400*stats::runif(100), "weeks")
cut(as.Date("2001/1/1") + 70*stats::runif(100), "weeks")

# The standards all have midnight as the start of the day, but some
# people incorrectly interpret it at the end of the previous day ...
tm <- seq(as.POSIXct("2012-06-01 06:00"), by = "6 hours", length.out = 24)
aggregate(1:24, list(day = cut(tm, "days")), mean)
# and a version with midnight included in the previous day:
aggregate(1:24, list(day = cut(tm, "days", right = TRUE)), mean)

```

data.class

```
data.class(x)
```

```
x
```

```

x
classNULLdimNULLmode(x)
data.class(x)

```



```
xintegerdata.class(x)"numeric"x
```

class

```
x <- LETTERS
data.class(factor(x))           # has a class attribute
data.class(matrix(x, ncol = 13)) # has a dim attribute
data.class(list(x))             # the same as mode(x)
data.class(x)                   # the same as mode(x)

stopifnot(data.class(1:2) == "numeric") # compatibility "rule"
```

data.frame

data.frame()

```
data.frame(..., row.names = NULL, check.rows = FALSE,
            check.names = TRUE, fix.empty.names = TRUE,
            stringsAsFactors = FALSE)
```

```
...          valuetag = value
row.names    NULL
check.rows   TRUE
check.names  TRUEmake.names
fix.empty.names
            someName = arg""FALSEcheck.names""
stringsAsFactors
            TRUEFALSE
```

"data.frame"

```
check.names = FALSEdata.frame
```

```
data.frameas.data.frame(optional = TRUE)data.frameIstringsAsFactorsdata.frameI
```

```
data.frameis.vectorI
```

```
data.framerow.namesNULLas.matrix
```

```
row.names
```

```
I
```

```
I(... )check.names = FALSE
```

row.namesrow.names

[Iplot.data.frameprint.data.frame](#)[row.namesnames\[.data.frameI\(matrix\(..\)\)](#)
[Math.data.frame](#)[data.frame](#)[read.table](#)[make.nameslist2DF](#)

```
L3 <- LETTERS[1:3]
char <- sample(L3, 10, replace = TRUE)
(d <- data.frame(x = 1, y = 1:10, char = char))
## The "same" with automatic column names:
data.frame(1, 1:10, sample(L3, 10, replace = TRUE))

is.data.frame(d)

## enable automatic conversion of character arguments to factor columns:
(dd <- data.frame(d, fac = letters[1:10], stringsAsFactors = TRUE))
rbind(class = sapply(dd, class), mode = sapply(dd, mode))

stopifnot(1:10 == row.names(d)) # {coercion}

(d0 <- d[, FALSE]) # data frame with 0 columns and 10 rows
(d.0 <- d[FALSE, ]) # <0 rows> data frame (3 named cols)
(d00 <- d0[FALSE, ]) # data frame with 0 columns and 0 rows
```

data.matrix

data.matrix(frame, rownames.force = NA)

frame

rownames.force NULL[rownames](#)[NANULL](#)

[is.numeric](#)[as.numeric](#)(, "numeric")

frame"data.frame"frame[row.names](#)[NULL](#)[rownames](#)[.force](#)[names](#)
[as.matrix](#)

[as.matrixdata.frame](#)[matrix](#)

```
DF <- data.frame(a = 1:3, b = letters[10:12],
                 c = seq(as.Date("2004-01-01"), by = "week", length.out = 3),
                 stringsAsFactors = TRUE)
data.matrix(DF[1:2])
data.matrix(DF)
```

date

`date()`

"Fri Aug 20 11:11:00 1999"ctime

[Sys.Date](#)[Sys.time](#)[Date](#)[Date](#)[Time](#)[Classes](#)

```
(d <- date())
nchar(d) == 24

## something similar in the current locale
## depending on ctime; e.g. %e could be %d:
format(Sys.time(), "%a %b %e %H:%M:%S %Y")
```

Dates

"Date"

```
## S3 method for class 'Date'  
summary(object, digits = 12, ...)
```

```
## S3 method for class 'Date'  
print(x, max = NULL, ...)
```

```
objectx      Date  
digits  
max          NULLNULLgetOption("max.print")  
...
```

```
meanOps.Date  
as.POSIXctas.POSIXlt  
"POSIXlt"NA  
methods(class = "Date")
```

```
identical()"Date"storage.mode"double""integer"
```

```
Sys.Date  
weekdays
```

```
Ops.Date "Date"  
format.Date  
axis.Date hist.Date  
seq.Date cut.Dateround.Date  
  
DateTimeClasses
```

```

(today <- Sys.Date())
format(today, "%d %b %Y") # with month as a word
(tenweeks <- seq(today, length.out=10, by="1 week")) # next ten weeks
weekdays(today)
months(tenweeks)

(Dls <- as.Date(.leap.seconds))

## Show use of year zero:
(z <- as.Date("01-01-01")) # how it is printed depends on the OS
z - 365 # so year zero was a leap year.
as.Date("00-02-29")
# if you want a different format, consider something like (if supported)
## Not run: format(z, "%04Y-%m-%d") # "0001-01-01"
format(z, "%_4Y-%m-%d") # " 1-01-01"
format(z, "%_Y-%m-%d") # "1-01-01"

## End(Not run)

## length(<Date>) <- n now works
ls <- Dls; length(ls) <- 12
l2 <- Dls; length(l2) <- 5 + length(Dls)
stopifnot(exprs = {
  ## length(.) <- * is compatible to subsetting/indexing:
  identical(ls, Dls[seq_along(ls)])
  identical(l2, Dls[seq_along(l2)])
  ## has filled with NA's
  is.na(l2[(length(Dls)+1):length(l2)])
})

```

DateTimeClasses

"POSIXlt" "POSIXct"

```

## S3 method for class 'POSIXct'
print(x, tz = "", usetz = TRUE, max = NULL,
      digits = getOption("digits.secs"), ...)

```

```

## S3 method for class 'POSIXct'
summary(object, digits = 15, ...)

```

```

time + z
z + time
time - z
time1 lop time2

```

.leap.seconds

```

xobject
tzusetz      format.POSIXct
max          NULLNULLgetOption("max.print")
digits       print()
...
time
time1time2   as.POSIXct
z
lop          ==!=<=>=>=

```

```

"POSIXct""POSIXlt"listsecminhourmdaymonyearwdayydayisdstzonegmttoff
time_tstruct tm
"POSIXct""POSIXlt""POSIXt"
difftime"POSIXlt"
meansummaryuniqueis.finitelength<-prettyhistAxismethods(class = "POSIXt")..
class = "POSIXlt""POSIXct"
"tzzone"
.leap.seconds"POSIXct"
c"POSIXlt""POSIXct""tzzone"
"2005-12-31 23:59:60""1969-12-31 23:59:59 UTC"-1as.POSIXct("1969-12-31
23:59:59", format = "%Y-%m-%d %H:%M:%S", tz = "UTC")NAas.POSIXct("1969-12-31
23:59:59", tz = "UTC")"1969-12-31 23:59:00"
options("max.print")

```

```

"POSIXlt""tzzone"TZtz"POSIXlt""
"POSIXct""tzzone""POSIXlt""tzzone"

```

```

"POSIXlt"list
sec
min
hour
mday
mon
year
wday
yday
isdst
zone "" "" ""

```

gmtoff NA0

gmtoffglibctzcode

length()print()str()"POSIXct"

"POSIXlt"integersecdoublezonecharacterzone

wdayydayformat()as.character()

isdstisdst1

"POSIXlt"balancePOSIXlt()

"POSIXct""POSIXlt"

options("digits.secs")strftime

"POSIXlt"

POSIXctlong

isdst-1

TZTZSys.timezone

"POSIXlt"

"POSIXlt"

identical()"POSIXct"==all.equalstorage.mode"double""integer"

https://www.r-project.org/doc/Rnews/Rnews_2001-2.pdf

as.POSIXctas.POSIXlt

strptime

Sys.time"POSIXct"

difftime

balancePOSIXlt()

cut.POSIXtseq.POSIXtround.POSIXtrunc.POSIXt

weekdaysmonths()quarters()julian()

```

(z <- Sys.time())           # the current date, as class "POSIXct"

Sys.time() - 3600           # an hour ago

as.POSIXlt(Sys.time(), "GMT") # the current time in GMT
format(.leap.seconds)        # the leap seconds in your time zone
print(.leap.seconds, tz = "America/Los_Angeles") # and in Seattle's

## look at *internal* representation of "POSIXlt" :
leapS <- as.POSIXlt(.leap.seconds)
names(unclass(leapS)) ; is.list(leapS)
## str() on inner structure needs unclass(.):
utils::str(unclass(leapS), vec.len = 7)
## show all (apart from "tzone" attr):
data.frame(unclass(leapS))

## Extracting *single* components of POSIXlt objects:
leapS[1 : 5, "year"]
leapS[17:22, "mon" ]

```

dcf

```
read.dcf(file, fields = NULL, all = FALSE, keep.white = NULL)
```

```
write.dcf(x, file = "", append = FALSE, useBytes = FALSE,
          indent = 0.1 * getOption("width"),
          width = 0.9 * getOption("width"),
          keep.white = NULL)
```

file	""read.dcfgzfile
fields	
all	all
keep.white	NULLread.dcfwrite.dcfstrwrap
x	x
append	TRUEFALSE
useBytes	writeLines()
indent	
width	


```
tag:value::
```

```
.
```

```
read.dcf(all = FALSE)DESCRIPTIONEncoding  
write.dcfNA
```

```
read.dcf(all = FALSE)keep.whitefieldsNA  
read.dcf(all = TRUE)  
fileread.dcf  
write.dcfNULL
```

```
https://www.debian.org/doc/debian-policy/ch-controlfields.html
```

```
#
```

```
write.table
```

```
available.packagesread.dcf
```

```
## Create a reduced version of the DESCRIPTION file in package 'splines'  
x <- read.dcf(file = system.file("DESCRIPTION", package = "splines"),  
              fields = c("Package", "Version", "Title"))  
write.dcf(x)
```

```
## An online DCF file with multiple records  
con <- url("https://cran.r-project.org/src/contrib/PACKAGES")  
y <- read.dcf(con, all = TRUE)  
close(con)  
utils::str(y)
```

debug

textconditionbrowser

```
debug(fun, text = "", condition = NULL, signature = NULL)
debugonce(fun, text = "", condition = NULL, signature = NULL)
undebug(fun, signature = NULL)
isdebugged(fun, signature = NULL)
debuggingState(on = NULL)
```

fun

text

condition

signature

on debuggingStateTRUEFALSEFALSEbrowser

browser

browser

debug

trace(..., at = *)setBreakpoint

debugdebugonce()

signaturefuntextconditionsignature.local.localisRematched

isdebuggedTRUEsignatureNULLfunsignatureNULLfunfunFALSE

options(deparse.max.lines)

debugundebugNULL

isdebuggedTRUE

FALSE

debugcallbrowsertracetracebackError: ...recover

```
## Not run:
debug(library)
library(methods)

## End(Not run)
## Not run:
debugonce(sample)
## only the first call will be debugged
sample(10, 1)
sample(10, 1)

## End(Not run)
```

declare

```
declare(...)
```

```
...
```

```
declare()NULL
```

Defunct

```
.Defunct
```

```
.Defunct(new, package = NULL, msg)
```

```
new
package
msg
```

```
.Defuncthelp("--defunct")
.Defunct"defunctError"oldnewpackage
```

[Deprecated](#)

```
help("base-defunct")
```

delayedAssign

delayedAssign

```
delayedAssign(x, value, eval.env = parent.frame(1),
              assign.env = parent.frame(1))
```

```
x
value      x
eval.env   value
assign.env x
```

```
eval.envassign.env
delayedAssign
eval.env
```

valuex

[substitute](#)assign.env.GlobalEnv

```
msg <- "old"
delayedAssign("x", msg)
substitute(x) # shows only 'x', as it is in the global env.
msg <- "new!"
x # new!
```

```
delayedAssign("x", {
  for(i in 1:3)
    cat("yippee!\n")
  10
})
```

```
x^2 #- yippee
x^2 #- simple number
```

```
ne <- new.env()
delayedAssign("x", pi + 2, assign.env = ne)
## See the promise {without "forcing" (i.e. evaluating) it}:
substitute(x, ne) # 'pi + 2'
```

Promises in an environment [for advanced users]: -----

```

e <- (function(x, y = 1, z) environment())(cos, "y", {cat(" HO!\n"); pi+2})
## How can we look at all promises in an env (w/o forcing them)?
gete <- function(e_) {
  ne <- names(e_)
  names(ne) <- ne
  lapply(lapply(ne, as.name),
    function(n) eval(substitute(substitute(X, e_), list(X=n))))
}
(exps <- gete(e))
sapply(exps, typeof)

(1e <- as.list(e)) # evaluates ("force"s) the promises
stopifnot(identical(1e, lapply(exps, eval))) # and another "Ho!"

```

deparse

```

deparse(expr, width.cutoff = 60L,
  backtick = mode(expr) %in% c("call", "expression", "(", "function"),
  control = c("keepNA", "keepInteger", "niceNames", "showAttributes"),
  nlines = -1L)

```

```

deparse1(expr, collapse = " ", width.cutoff = 500L, ...)

```

```

expr
width.cutoff    [20,500]
backtick
control         NULLcontrol = "all".deparse0pts
nlines
collapse        paste()
...             deparse()

```

```

modetypeof"expression"expressionparse
deparsesubstitutemyplot
backtick
width.cutoffwidth.cutoffarg = value
deparse1()characterdeparse1(substitute(.))

```

```

recordPlot.Internal

```

[.deparseOpts](#)[control](#)[dput\(\)](#)[dump\(\)](#)

[substitute](#)[parse](#)[expression](#)

Quotes

```
require(stats); require(graphics)
```

```
deparse(args(lm))
```

```
deparse(args(lm), width.cutoff = 500)
```

```
myplot <- function(x, y) {  
  plot(x, y, xlab = deparse1(substitute(x)),  
       ylab = deparse1(substitute(y)))  
}
```

```
e <- quote(`foo bar`)
```

```
deparse(e)
```

```
deparse(e, backtick = TRUE)
```

```
e <- quote(`foo bar`+1)
```

```
deparse(e)
```

```
deparse(e, control = "all") # wraps it w/ quote( . )
```

[deparseOpts](#)

[deparsed](#)[dput](#)[dump](#)

[.deparseOpts](#)[\(control\)](#)

[..deparseOpts](#)

[control](#)

[..deparseOpts](#)[character](#)[.deparseOpts\(\)](#)

[.deparseOpts\(\)](#)[deparsed](#)[dput](#)[dump](#)[control](#)

[control](#)[..deparseOpts](#)

"keepInteger" [as.integer\(\)](#)[LNANA_integer_NA](#)"S_compatible"

"quoteExpressions" [formula](#)[quote\(\)](#)

"showAttributes" [attributes](#)[sources](#)[src](#)[ref](#)[structure\(\)](#)[names](#)"niceNames"

"showAttributes"[deparsed](#)[dput](#)

```

"useSource" sourcesrcref
"warnIncomplete"

"keepNA" NANA_real_S_compatible
"niceNames" listNAnamesc(A = 1)structure(1, names = "A")"showAttributes"
"all" "digits17"dump"digits17"edit
"delayPromises"
"S_compatible"
"hexNumeric" "%a"sprintf

"digits17" "%.17g"
"exact" control = c("all", "hexNumeric")

control = NULLdeparsedumpcontrol = "all" c("keepInteger", "keepNA")
"warnIncomplete"

control = "exact"control = c("all", "hexNumeric")deparse()parse()"all""digits17"
"hexNumeric""digits17"

control

stopifnot(.deparseOpts("exact") == .deparseOpts(c("all", "hexNumeric")))
(iOpt.all <- .deparseOpts("all")) # a four digit integer

## one integer --> vector binary bits
int2bits <- function(x, base = 2L,
                     ndigits = 1 + floor(1e-9 + log(max(x,1), base))) {
  r <- numeric(ndigits)
  for (i in ndigits:1) {
    r[i] <- x%%base
    if (i > 1L)
      x <- x/%base
  }
  rev(r) # smallest bit at left
}
int2bits(iOpt.all)
## What options does "all" contain ? =====
(dep0.indiv <- setdiff(.deparseOpts, c("all", "exact")))
(oa <- dep0.indiv[int2bits(iOpt.all) == 1])# 8 strings
stopifnot(identical(iOpt.all, .deparseOpts(oa)))

## ditto for "exact" instead of "all":
(iOpt.X <- .deparseOpts("exact"))
data.frame(opts = dep0.indiv,
           all = int2bits(iOpt.all),
           exact= int2bits(iOpt.X))
(oX <- dep0.indiv[int2bits(iOpt.X) == 1]) # 8 strings, too
diffXall <- oa != oX
stopifnot(identical(iOpt.X, .deparseOpts(oX)),
           identical(oX[diffXall], "hexNumeric"),
           identical(oa[diffXall], "digits17"))

```

Deprecated

`.Deprecated`

```
.Deprecated(new, package = NULL, msg,  
            old = as.character(sys.call(sys.parent()))[1L])
```

`new`
`package`
`msg`
`old`

```
.Deprecated("")help("-deprecated")help("-deprecated")  
.Deprecated"deprecatedWarning"oldnewpackage
```

[Defunct](#)

```
help("base-deprecated")
```

det

`detdeterminant`

```
det(x, ...)  
determinant(x, logarithm = TRUE, ...)
```

`x`
`logarithm` `TRUE`
`...`

`determinantdetdeterminant`

`detxdeterminant`

`modulus` `logarithmFALSE`
`sign` `+1-1`


```
(x <- matrix(1:4, ncol = 2))  
unlist(determinant(x))  
det(x)  
  
det(print(cbind(1, 1:3, c(2,0,1))))
```

detach

[search\(\)](#)[data.frame](#)[attach](#)[library](#)

```
detach(name, pos = 2L, unload = FALSE, character.only = FALSE,  
       force = FALSE)
```

name	search() [pos]pos
pos	search() namepos = name
unload	unloadTRUEdetach unloadNamespace unloadFALSE
character.only	name
force	

```
package:tools  
unload = TRUEgetLoadedDLLslibrary.dynam.unloadlibrary
```

NULL[attach](#)

```
detach()attach  
attachdetachattach
```

[attach](#)[library](#)[search](#)[objects](#)[unloadNamespace](#)[library.dynam.unload](#)

```

require(splines) # package
detach(package:splines)
## or also
library(splines)
pkg <- "package:splines"

detach(pkg, character.only = TRUE)

## careful: do not do this unless 'splines' is not already attached.
library(splines)
detach(2) # 'pos' used for 'name'

## an example of the name argument to attach
## and of detaching a database named by a character vector
attach_and_detach <- function(db, pos = 2)
{
  name <- deparse1(substitute(db))
  attach(db, pos = pos, name = name)
  print(search()[pos])
  detach(name, character.only = TRUE)
}
attach_and_detach(women, pos = 3)

```

diag

```

diag(x = 1, nrow, ncol, names = TRUE)
diag(x) <- value

```

x	array
nrowncol	x
names	xxnamesdimnames(x)
value	x

```

diag
  x
  xrow
  x
  xcomplexnumericintegerlogicalraw
nrowncol

```

```

xdiag(x)xnamesnamesx
x
nrowncolnrowxnrowncolnrow = as.integer(x)
x

diag(x)xdiag(x, nrow = length(x))

```

[upper.tril](#)[lower.tri](#)[matrix](#)

```

dim(diag(3))
diag(10, 3, 4) # guess what?
all(diag(1:3) == {m <- matrix(0,3,3); diag(m) <- 1:3; m})

## other "numeric"-like diagonal matrices :
diag(c(1i,2i)) # complex
diag(TRUE, 3) # logical
diag(as.raw(1:3)) # raw
(D2 <- diag(2:1, 4)); typeof(D2) # "integer"

require(stats)
## diag(<var-cov-matrix>) = variances
diag(var(M <- cbind(X = 1:5, Y = rnorm(5))))
#-> vector with names "X" and "Y"
rownames(M) <- c(colnames(M), rep("", 3))
M; diag(M) # named as well
diag(M, names = FALSE) # w/o names

```

diff

```

diff(x, ...)

## Default S3 method:
diff(x, lag = 1, differences = 1, ...)

## S3 method for class 'POSIXt'
diff(x, lag = 1, differences = 1, ...)

## S3 method for class 'Date'
diff(x, lag = 1, differences = 1, ...)

```

```
x
lag
differences
...
```

```
diff"ts""POSIXt""Date"
NA
```

```
xndifferences = 1x[(1+lag):n] - x[1:(n-lag)]
differencecexx
x
```

```
diff.tsdiffinv
```

```
diff(1:10, 2)
diff(1:10, 2, 2)
x <- cumsum(cumsum(1:10))
diff(x, lag = 2)
diff(x, differences = 2)

diff(.leap.seconds)
## allows to pass units via ... to difftime()
diff(.leap.seconds, units = "weeks")
diff(as.Date(.leap.seconds), units = "weeks")
```

```
difftime
```

```
print()
```

```
time1 - time2
```

```
difftime(time1, time2, tz,
          units = c("auto", "secs", "mins", "hours",
                    "days", "weeks"))
```

```
as.difftime(tim, format = "%X", units = "auto", tz = "UTC")
```

```
## S3 method for class 'difftime'
```

```

format(x, ..., with.units = TRUE)
## S3 method for class 'difftime'
units(x)
## S3 replacement method for class 'difftime'
units(x) <- value
## S3 method for class 'difftime'
as.double(x, units = "auto", ...)

## Group methods, notably for round(), signif(), floor(),
## ceiling(), trunc(), abs(); called directly, *not* as Math():
## S3 method for class 'difftime'
Math(x, ...)

```

```

time1time2
tz          "POSIXlt"
units
value      units
tim
format     timstrptime
x          "difftime"
...
with.units format()

```

```

difftime1time2"difftime"MathroundsigniffloorceilingtruncabssignOps
units = "auto""weeks"
difftimeunits = "auto"as.difftime()format
"difftime""difftime""difftime""difftime"meansumSummarydiffdiff.default
"difftime"-
"difftime"units
units = "days""Date"
as.doubleunits = "auto"
format

```

```

z <- as.POSIXct(c("2016-12-31 23:59:59", "2017-01-01 00:00:01"))
z[2] - z[1]

```

Time difference of 2 secs

"months"seq.Dateseq.POSIXt

DateTimeClasses

```
(z <- Sys.time() - 3600)
Sys.time() - z           # just over 3600 seconds.

## time interval between release days of R 1.2.2 and 1.2.3.
ISOdate(2001, 4, 26) - ISOdate(2001, 2, 26)

as.difftime(c("0:3:20", "11:23:15"))
as.difftime(c("3:20", "23:15", "2:"), format = "%H:%M") # 3rd gives NA
(z <- as.difftime(c(0,30,60), units = "mins"))
as.numeric(z, units = "secs")
as.numeric(z, units = "hours")
format(z)
```

dim

```
dim(x)
dim(x) <- value
```

```
x
value          NULL
```

```
dimdim<-
dimdata.frame row.namesxx
```

```
dimdimNULLinteger
"dim""dimnames""names"
```

ncolnrowdimnames

```
x <- 1:12 ; dim(x) <- c(3,4)
x
```

```
# simple versions of nrow and ncol could be defined as follows
nrow0 <- function(x) dim(x)[1]
ncol0 <- function(x) dim(x)[2]
```

dimnames

```
dimnames(x)
dimnames(x) <- value

provideDimnames(x, sep = "", base = list(LETTERS), unique = TRUE)
```

```
x
value      dimnames(x)
sep        base
base       list
unique     make.unique
```

```
dimnamesdimnames<-
arraymatrixdimnamesvalue
valueas.characterNULLNULLvalueNULL
row.namesnamesvalueas.character
namesdimnames
```

```
provideDimnames(x)dimnamescharacteruniquemake.unique(*, sep=sep)
```

```
NULLdim(x)NULLxNULLNULL
"data.frame"
provideDimnames(x)xNULLdimnames
```

```
dimnames(A)[[1]] <- valueNULLrownames
```

```
rownamescolnamesarraymatrixdata.frame
```

```
## simple versions of rownames and colnames
## could be defined as follows
rownames0 <- function(x) dimnames(x)[[1]]
colnames0 <- function(x) dimnames(x)[[2]]

(dn <- dimnames(A <- provideDimnames(N <- array(1:24, dim = 2:4))))
A0 <- A; dimnames(A)[2:3] <- list(NULL)
stopifnot(identical(A0, provideDimnames(A)))
strd <- function(x) utils::str(dimnames(x))
strd(provideDimnames(A, base= list(letters[-(1:9)], tail(LETTERS))))
strd(provideDimnames(N, base= list(letters[-(1:9)], tail(LETTERS)))) # recycling
strd(provideDimnames(A, base= list(c("AA", "BB")))) # recycling on both levels
## set "empty dimnames":
provideDimnames(rbind(1, 2:3), base = list(""), unique=FALSE)
```

do.call

do.call

do.call(what, args, quote = FALSE, envir = parent.frame())

what

args namesargs

quote

envir what

quoteFALSEenvirquoteTRUEquote

substitutedo.call

.Internal

call


```

do.call("complex", list(imaginary = 1:3))

## if we already have a list (e.g., a data frame)
## we need c() to add further arguments
tmp <- expand.grid(letters[1:2], 1:3, c("+", "-"))
do.call("paste", c(tmp, sep = ""))

do.call(paste, list(as.name("A"), as.name("B")), quote = TRUE)

## examples of where objects will be found.
A <- 2
f <- function(x) print(x^2)
env <- new.env()
assign("A", 10, envir = env)
assign("f", f, envir = env)
f <- function(x) print(x)
f(A) # 2
do.call("f", list(A)) # 2
do.call("f", list(A), envir = env) # 4
do.call(f, list(A), envir = env) # 2
do.call("f", list(quote(A)), envir = env) # 100
do.call(f, list(quote(A)), envir = env) # 10
do.call("f", list(as.name("A")), envir = env) # 100

eval(call("f", A)) # 2
eval(call("f", quote(A))) # 2
eval(call("f", A), envir = env) # 4
eval(call("f", quote(A)), envir = env) # 100

```

dontCheck

dontCheckidentityR CMD checkxcheckFF(registration = TRUE).NAME

dontCheck(x)

x

suppressForeignCheckdontCheck

dots	..1
------	-----

```
.....1..2...
...elt(n)..eval(paste0("..", n))switch(n, ...)NULLn
...length().....names()nameslength(list(...))names(list(...))...list(...)
.....1..2...elt(n)
```

```
...length()
...names()
...elt(n)
```

```
n          ...length()length(list(...))...
```

```
.....1..2Reserved
```

```
tst <- function(n, ...) ...elt(n)
tst(1, pi=pi*0:1, 2:4) ## [1] 0.000000 3.141593
tst(2, pi=pi*0:1, 2:4) ## [1] 2 3 4
try(tst(1)) # -> Error about '...' not containing an element.

tst.dl <- function(x, ...) ...length()
tst.dns <- function(x, ...) ...names()
tst.dl(1:10) # 0 (because the first argument is 'x')
tst.dl(4, 5) # 1
tst.dl(4, 5, 6) # 2 namely '5, 6'
tst.dl(4, 5, 6, 7, sin(1:10), "foo"/"bar") # 5. Note: no evaluation!
tst.dns(4, foo=5, 6, bar=7, sini = sin(1:10), "foo"/"bar")
## "foo" "" "bar" "sini" ""

## From R 4.1.0 to 4.1.2, ...names() sometimes did not match names(list(...));
## check and show (these examples all would've failed):
chk.n2 <- function(...) stopifnot(identical(print(...names()), names(list(...))))
chk.n2(4, foo=5, 6, bar=7, sini = sin(1:10), "bar")
chk.n2()
chk.n2(1,2)
```

double

```
double(length = 0)
as.double(x, ...)
is.double(x)
```

```
single(length = 0)
as.single(x, ...)
```

length

x

...

double@numeric

as.doubleas.numeric"double"

is.double

as.singlesingleas.doubledoubleCsingle.C.Fortran

double@

as.doubleas.vectorstorage.mode0x0X"NA""NaN""Inf""infinity"

as.doublefactor

is.doubleTRUEFALSE

2×10^{-308} 2×10^{308} NaN

.Machine

doublenumericreal

doublenumeric"numeric"

"numeric"is.numericas.numericas.double

https://en.wikipedia.org/wiki/IEEE_754-1985https://en.wikipedia.org/wiki/IEEE_754-2008https://en.wikipedia.org/wiki/IEEE_754-2019https://en.wikipedia.org/wiki/Double_precisionhttps://en.wikipedia.org/wiki/Denormal_number

[integernumericstorage.mode](#)

```
is.double(1)
all(double(3) == 0)
```

dput

```
dput(x, file = "",
      control = c("keepNA", "keepInteger", "niceNames", "showAttributes"))
```

```
dget(file, keep.source = FALSE)
```

```
x
file      ""
control    NULLcontrol = "all".deparseOpts
keep.source
```

```
dputfilexdumpx
controldput()dumpcontrol = "all"control = NULL
dput
"useSource"controloptions(keep.source = FALSEsource
```

dput

dget

[dumpsavesaveRDS](#)dput

[deparse.deparseOptsdumpwrite](#)

```

fil <- tempfile()
## Write an ASCII version of the 'base' function mean() to our temp file, ..
dput(base::mean, fil)
## ... read it back into 'bar' and confirm it is the same
bar <- dget(fil)
stopifnot(all.equal(bar, base::mean, check.environment = FALSE))

## Create a function with comments
baz <- function(x) {
  # Subtract from one
  1-x
}
## and display it
dput(baz)
## and now display the saved source
dput(baz, control = "useSource")

## Numeric values:
xx <- pi^(1:3)
dput(xx)
dput(xx, control = "digits17")
dput(xx, control = "hexNumeric")
dput(xx, fil); dget(fil) - xx # slight rounding on all platforms
dput(xx, fil, control = "digits17")
dget(fil) - xx # slight rounding on some platforms
dput(xx, fil, control = "hexNumeric"); dget(fil) - xx
unlink(fil)

xn <- setNames(xx, paste0("pi^",1:3))
dput(xn) # nicer, now "niceNames" being part of default 'control'
dput(xn, control = "S_compat") # no names
## explicitly asking for output as in R < 3.5.0:
dput(xn, control = c("keepNA", "keepInteger", "showAttributes"))

```

drop

drop(x)

x

`xdimarray``drop``xdimnames``xdimnames``xdimnames``names``dimnames``names``dimnames``NULL`

`[drop = FALSE``drop`

`drop1``droplevels``factor`

```

dim(drop(array(1:12, dim = c(1,3,1,1,2,1,2)))) # = 3 2 2
drop(1:3 %*% 2:4) # scalar product - w/o drop(.), would return 1x1 matrix

# Behavior when result is length-1 vector:
(x <- x1 <- x2 <- array(0, c(1L, 1L), list("a", "b")))
colnames(x1) <- rownames(x2) <- NULL
names(drop(x )) # NULL
names(drop(x1)) # "a"
names(drop(x2)) # "b"

```

droplevels

droplevels[factor](#)

```

droplevels(x, ...)
## S3 method for class 'factor'
droplevels(x, exclude = if(anyNA(levels(x))) NULL else NA, ...)
## S3 method for class 'data.frame'
droplevels(x, except, exclude, ...)

```

```

x
exclude      factor\(\)NANaxx[ , drop=TRUE]
...
except

```

```

"factor"factor(x, exclude=exclude)excludeexclude
except

```

droplevelsx

except

[subsetfactor](#)[dropdrop1\[.factor](#)

```

aq <- transform(airquality, Month = factor(Month, labels = month.abb[5:9]))
aq <- subset(aq, Month != "Jul")
table(aq $Month)
table(droplevels(aq)$Month)

```

dump

dumpsource

```
dump(list, file = "dumpdata.R", append = FALSE,  
      control = "all", envir = parent.frame(), evaluate = TRUE)
```

```
list          NULL  
file          ""  
append        TRUEfilefilefile  
control        NULL.deparseOpts  
envir  
evaluate
```

```
file  
source  
dump  
dump  
dumpsource  
savesaveRDS  
dumpcontrol = "exact"  
control = NULLsource.deparseOpts  
control = c("keepInteger", "warnIncomplete", "keepNA")  
evaluate = TRUEdelayedAssignevaluate = FALSE
```

dumpdumpenvir

S4sourcecontrol

```
.deparseOptscontrol  
dput()dget()deparse()  
write  
write.table  
savesaveRDS
```

```
x <- 1; y <- 1:10  
fil <- tempfile(fileext=".Rdmped")  
dump(ls(pattern = '^[xyz]'), fil)  
print(.Last.value)  
unlink(fil)
```

duplicated

```
duplicated()
anyDuplicated(.)any(duplicated(.))TRUE

duplicated(x, incomparables = FALSE, ...)

## Default S3 method:
duplicated(x, incomparables = FALSE,
           fromLast = FALSE, nmax = NA, ...)

## S3 method for class 'array'
duplicated(x, incomparables = FALSE, MARGIN = 1,
           fromLast = FALSE, ...)

anyDuplicated(x, incomparables = FALSE, ...)
## Default S3 method:
anyDuplicated(x, incomparables = FALSE,
              fromLast = FALSE, ...)
## S3 method for class 'array'
anyDuplicated(x, incomparables = FALSE,
              MARGIN = 1, fromLast = FALSE, ...)

x          is.vectorNULL
incomparables FALSEx
fromLast    duplicated = FALSE
nmax
...
MARGIN      applyMARGIN = 0

duplicatedanyDuplicatedanyDuplicated(x,    ...)any(duplicated(x,    ...))ix[i]0
duplicatedanyDuplicated
duplicated(x, fromLast = TRUE)rev(duplicated(rev(x)))
MARGINfromLast = TRUEMARGIN = 2MARGIN = 0x
"NA"NaNmatchunique
incomparables
nmax = NANmax = length(x)8*nmaxnmaxlength(x)NAnmaxnmax = 1
duplicatednmax
```



```
duplicated()xMARGIN = 0
anyDuplicated()0
```

[vector](#) $O(n^2)$

[unique](#)

```
x <- c(9:20, 1:5, 3:7, 0:8)
## extract unique elements
(xu <- x[!duplicated(x)])
## similar, same elements but different order:
(xu2 <- x[!duplicated(x, fromLast = TRUE)])

## xu == unique(x) but unique(x) is more efficient
stopifnot(identical(xu, unique(x)),
           identical(xu2, unique(x, fromLast = TRUE)))

which(duplicated(warpbreaks))

## for 3d array, MARGIN = -1 <==> MARGIN = 2:3
duplicated(gait, MARGIN = -1) # boy26 duplicates boy19, see help(gait).
anyDuplicated(warpbreaks)

anyDuplicated(x)
anyDuplicated(x, fromLast = TRUE)
```

[dyn.load](#)

```
dyn.load(x, local = TRUE, now = TRUE, ...)
dyn.unload(x)
```

```
is.loaded(symbol, PACKAGE = "", type = "")
```

```
x
local
now
...
symbol
PACKAGE      name.so.sl.dll.C.Call.Fortran.External
type         ""Fortran""Call""External"
```

`dyn.load`

`PACKAGE`

`dyn.loadaddlopen()`

`local`

`now = FALSETRUEFALSE`

`_init()`

`verbosewarnoptions`

<https://www.stat.ucdavis.edu/~duncan/R/dynload/>

`dyn.load.C.Call.Fortran.Externaldyn.loadDLLInfogetLoadedDLLs`

`dyn.unload`

`is.loaded.NAME.C.Fortran.Call.Externaltype.Fortran`

`dyn.unloadlibrary.dynamlibrary.dynam.unload`

`is.loaded.C`

`R_MAX_NUM_DLLSulimit -nshbashlimit descriptorscsR_MAX_NUM_DLLS`

`R_MAX_NUM_DLLS`

`dlopenaddlopenLoadLibrary`

`library.dynam.onLoad`

`SHLIB`

`.C.Fortran.External.Call`

`## expect all of these to be false in R >= 3.0.0 as these can only be`

`## used via registered symbols.`

`is.loaded("supsmu") # Fortran entry point in stats`

`is.loaded("supsmu", "stats", "Fortran")`

`is.loaded("PDF", type = "External") # pdf() device in grDevices`

eapply

eapplyFUNenvironment

```
eapply(env, FUN, ..., all.names = FALSE, USE.NAMES = TRUE)
```

```
env
FUN      match.fun+%%*%
...      FUN
all.names
USE.NAMES names
```

```
USE.NAMES = FALSE
```

environmentlapply

```
require(stats)

env <- new.env(hash = FALSE) # so the order is fixed
env$a <- 1:10
env$beta <- exp(-3:3)
env$logic <- c(TRUE, FALSE, FALSE, TRUE)
# what have we there?
utils::ls.str(env)

# compute the mean for each list element
  eapply(env, mean)
unlist(eapply(env, mean, USE.NAMES = FALSE))

# median and quartiles for each element (making use of "... " passing):
eapply(env, quantile, probs = 1:3/4)
eapply(env, quantile)
```

eigen

```
eigen(x, symmetric, only.values = FALSE, EISPACK = FALSE)
```

```
x
symmetric      TRUEsymmetricisSymmetric(x)
only.values    TRUE
EISPACK
```

```
symmetricisSymmetric(x)
```

```
xx
1
NaNx
```

```
x
values          pxMod(values)
vectors          $p \times p$ NULLonly.valuesTRUE
```

```
only.values"eigen"
r <- eigen(A)V <- r$vectors; lam <- r$values

$$A = V\Lambda V^{-1}$$

 $\Lambda = \text{diag}(\text{lam})$ 
```

```
eigenDSYEVZGEEVZHEEVZGEEV
https://netlib.org/lapack/
```

```
https://netlib.org/lapack/lug/lapack\_lug.html
```

```
svdeigenqrchol
qrdet
```

```
eigen(cbind(c(1,-1), c(-1,1)))
eigen(cbind(c(1,-1), c(-1,1)), symmetric = FALSE)
# same (different algorithm).
```

```
eigen(cbind(1, c(1,-1)), only.values = TRUE)
eigen(cbind(-1, 2:1)) # complex values
eigen(print(cbind(c(0, 1i), c(-1i, 0)))) # Hermite ==> real Eigenvalues
## 3 x 3:
eigen(cbind( 1, 3:1, 1:3))
eigen(cbind(-1, c(1:2,0), 0:2)) # complex values
```

encodeString

encodeStringprint.default

```
encodeString(x, width = 0, quote = "", na.encode = TRUE,  
             justify = c("left", "right", "centre", "none"))
```

```
x                as.character  
width            NULLNAx  
quote  
na.encode        NA  
justify          justify == "none"width = 0format.default
```

```
\a\b\f\n\r\t\v\xyz
```

```
print.default
```

```
quote
```

```
x
```

```
widthformat.default
```

```
print.default
```

```
x <- "ab\bc\ndef"  
print(x)  
cat(x) # interprets escapes  
cat(encodeString(x), "\n", sep = "") # similar to print()
```

```
factor(x) # makes use of this to print the levels
```

```
x <- c("a", "ab", "abcde")  
encodeString(x) # width = 0: use as little as possible  
encodeString(x, 2) # use two or more (left justified)  
encodeString(x, width = NA) # left justification  
encodeString(x, width = NA, justify = "c")  
encodeString(x, width = NA, justify = "r")  
encodeString(x, width = NA, quote = "'", justify = "r")
```

Encoding

```
Encoding(x)
```

```
Encoding(x) <- value
```

```
enc2native(x)
```

```
enc2utf8(x)
```

```
x
```

```
value
```

```
"latin1""UTF-8""bytes"Encoding"latin1""UTF-8""bytes""unknown"value"unknown"
```

```
"bytes"writeLines(useBytes = TRUE)
```

```
enc2nativeenc2utf8
```

```
\u\Uscanread.tablereadLinesparseencodingiconvtointToUtf8"UTF-8"textConnection  
source(encoding=)
```

```
chartrstrsplit(useBytes = FALSE)tolowertouppergsub(useBytes = FALSE)gsub(useBytes  
= FALSE)
```

```
substrchartrtolowertoupperfixedperlstrsplitgsub
```

```
pastesprintf
```

```
matchpmatchcharmatchduplicatedunique
```

```
Sys.setlocale
```

```
enc2utf8enc2native
```

```
## x is intended to be in latin1
```

```
x. <- x <- "fran\xE7ais"
```

```
Encoding(x.) # "unknown" (UTF-8 loc.) | "latin1" (8859-1/CP-1252 loc.) | ....
```

```
Encoding(x) <- "latin1"
```

```
x
```

```
xx <- iconv(x, "latin1", "UTF-8")
```

```
Encoding(c(x., x, xx))
```

```
c(x, xx)
```

```
xb <- xx; Encoding(xb) <- "bytes"
```

```
xb # will be encoded in hex
```

```
cat("x = ", x, ", ", xx = ", xx, ", ", xb = ", xb, "\n", sep = "")
```

```
(Ex <- Encoding(c(x.,x,xx,xb)))
```

```
stopifnot(identical(Ex, c(Encoding(x.), Encoding(x),  
Encoding(xx), Encoding(xb))))
```

environment

```
environment(fun = NULL)
environment(fun) <- value

is.environment(x)

.GlobalEnv
globalenv()
.BaseNamespaceEnv

emptyenv()
baseenv()

new.env(hash = TRUE, parent = parent.frame(), size = 29L)

parent.env(env)
parent.env(env) <- value

environmentName(env)

env.profile(env)
```

```
fun          functionformulaNULL
value
x
hash          TRUE
parent
env
size          sizeNAhashFALSE
```

```
parent.frameparent.env
getexistsinherits = TRUE
.GlobalEnvglobalenv()
.BaseNamespaceEnvbaseenv()
parent.envemptyenv()
parent.env<-
environmentis.environmentbaseenvemptyenvglobalenv
attach()"name"
```

```

funenvironment(fun)funNULL
funvalueprimitivefunNULL
is.environment(obj)TRUEobjenvironment
new.env
parent.env
parent.env<-
environmentName""
env.profile$size$chains$HASHPRIcounts$envNULL

```

https://en.wikipedia.org/wiki/Hash_table

```

envirevalgetexists
lsls.str
sys.source

```

```

f <- function() "top level function"

##-- all three give the same:
environment()
environment(f)
.GlobalEnv

ls(envir = environment(stats::approxfun(1:2, 1:2, method = "const")))

is.environment(.GlobalEnv) # TRUE

e1 <- new.env(parent = baseenv()) # this one has enclosure package:base.
e2 <- new.env(parent = e1)
assign("a", 3, envir = e1)
ls(e1)
ls(e2)
exists("a", envir = e2) # this succeeds by inheritance
exists("a", envir = e2, inherits = FALSE)
exists("+", envir = e2) # this succeeds by inheritance

eh <- new.env(hash = TRUE, size = NA)
with(env.profile(eh), stopifnot(size == length(counts)))

```



```

HOME
LANGUAGE
LC_ALL Sys.getlocale
MAKEINDEX makeindextexi2dvitexi2pdf
R_BATCH R CMD BATCH""!is.na(Sys.getenv("R_BATCH", NA))
R_BROWSER options("browser")
R_COMPLETION FALSE
R_DEFAULT_PACKAGES options
R_DOC_DIR doc
R_ENVIRON
R_GSCMD dev2bitmapbitmapembedFontssystem
R_HISTFILE
R_HISTSIZE
    readlinesavehistory
    RguiRconsole
R_HOME R.home
R_INCLUDE_DIR include
R_LIBS .libPaths
R_LIBS_SITE .libPaths
R_LIBS_USER .libPaths
R_PAPERSIZE options("papersize")pdfpostscript
R_PCRE_JIT_STACK_MAXSIZE grep
R_PDFVIEWER R CMD Rd2pdf
R_PLATFORM "--"R.Version
R_PROFILE
R_RD4PDF pdflatexRdR CMD Rd2pdf
R_SHARE_DIR share
R_TEXI2DVICMD texi2dviTEXI2DVI
    options("texi2dvi")texi2dvitexi2pdf
R_TIDYCMD tidyR CMD check_R_CHECK_RD_VALIDATE_RD2HTML_--as-cran
R_UNZIPCMD unzipoptions("unzip")
R_ZIPCMD zipzipR CMD INSTALL --build
TMPDIRTMPTEMP tempdirTMPDIRbuild
TZ Sys.timezone
TZDIR Sys.timezone
no_proxyhttp_proxyftp_proxy download.file

```

```
DISPLAY X11
EDITOR options("editor")
PAGER options("pager")less
R_PRINTCMD options("printcmd")postscript
R_SUPPORT_OLD_TARS support_old_tarsuntarTRUEtarxz
```

```
GSC R_GSCMD
R_USER HOME
```

```
Sys.getenvSys.setenv
gctorture
```

```
eval
```

```
eval(expr, envir = parent.frame(),
      enclos = if(is.list(envir) || is.pairlist(envir))
                 parent.frame() else baseenv())
evalq(expr, envir, enclos)
eval.parent(expr, n = 1)
local(expr, envir = new.env())
```

```
expr
envir      environmentexprNULLsys.call
enclos     envirenvirNULLbaseenv()
n
```

```
evalexprnenvirenvirparent.frame()eval
callexpression
evalqeval(quote(expr), ...)evalevalq
eval.parent(expr, n)eval(expr, parent.frame(n))
envirenclosexpr
envirNULLenvirenclos
localevalq
```

```
eval(x, data, parent.frame())
```

```
eval
```

```
expressionquotesys.frameparent.frameenvironment  
force
```

```
eval(2 ^ 2 ^ 3)
mEx <- expression(2^2^3); mEx; 1 + eval(mEx)
eval({ xx <- pi; xx^2}) ; xx

a <- 3 ; aa <- 4 ; evalq(evalq(a+b+aa, list(a = 1)), list(b = 5)) # == 10
a <- 3 ; aa <- 4 ; evalq(evalq(a+b+aa, -1), list(b = 5))          # == 12

ev <- function() {
  e1 <- parent.frame()
  ## Evaluate a in e1
  aa <- eval(expression(a), e1)
  ## evaluate the expression bound to a in e1
  a <- expression(x+y)
  list(aa = aa, eval = eval(a, e1))
}
tst.ev <- function(a = 7) { x <- pi; y <- 1; ev() }
tst.ev() #-> aa : 7,  eval : 4.14

a <- list(a = 3, b = 4)
with(a, a <- 5) # alters the copy of a from the list, discarded.

##
## Example of evalq()
##

N <- 3
env <- new.env()
assign("N", 27, envir = env)
## this version changes the visible copy of N only, since the argument
## passed to eval is '4'.
eval(N <- 4, env)
N
get("N", envir = env)
## this version does the assignment in env, and changes N only there.
evalq(N <- 5, env)
N
get("N", envir = env)
```

```
##
## Uses of local()
##

# Mutually recursive.
# gg gets value of last assignment, an anonymous version of f.

gg <- local({
  k <- function(y)f(y)
  f <- function(x) if(x) x*k(x-1) else 1
})
gg(10)
sapply(1:5, gg)

# Nesting locals: a is private storage accessible to k
gg <- local({
  k <- local({
    a <- 1
    function(y){print(a <- a+1);f(y)}
  })
  f <- function(x) if(x) x*k(x-1) else 1
})
sapply(1:5, gg)

ls(envir = environment(gg))
ls(envir = environment(get("k", envir = environment(gg))))
```

exists

```
exists(x, where = -1, envir = , frame, mode = "any",
       inherits = TRUE)

get0(x, envir = pos.to.env(-1L), mode = "any", inherits = TRUE,
     ifnotfound = NULL)
```

```
x
where
envir      where
frame      wheresys.frame(frame)
mode
inherits
ifnotfound get0(x, *)x
```

```

where search environments sys.frame env
x inherits TRUE x environment
inherits = TRUE
mode mode "numeric" "function" mode = "special"

```

```

exists():
get0(): get(x, *) exists(x, *) ifnotfound

```

```

get0()

    if (exists(myVarName, envir = myEnvir)) {
      r <- get(myVarName, envir = myEnvir)
      ## ... deal with r ...
    }

    if (!is.null(r <- get0(myVarName, envir = myEnvir))) {
      ## ... deal with r ...
    }

```

```

get has name missing file.exists

```

```

## Define a substitute function if necessary:
if(!exists("some.fun", mode = "function"))
  some.fun <- function(x) { cat("some.fun(x)\n"); x }
search()
exists("ls", 2) # true even though ls is in pos = 3
exists("ls", 2, inherits = FALSE) # false

## These are true (in most circumstances):
identical(ls, get0("ls"))
identical(NULL, get0(".foo.bar.")) # default ifnotfound = NULL (!)

```

`expand.grid`

```
expand.grid(..., KEEP.OUT.ATTRS = TRUE, stringsAsFactors = TRUE)
```

```
...
```

```
KEEP.OUT.ATTRS "out.attrs"  
stringsAsFactors
```

```
"out.attrs"predict
```

```
combn(n, m)nm
```

```
require(utils)
```

```
expand.grid(height = seq(60, 80, 5), weight = seq(100, 300, 50),  
             sex = c("Male", "Female"))
```

```
x <- seq(0, 10, length.out = 100)  
y <- seq(-1, 1, length.out = 20)  
d1 <- expand.grid(x = x, y = y)  
d2 <- expand.grid(x = x, y = y, KEEP.OUT.ATTRS = FALSE)  
object.size(d1) - object.size(d2)  
##-> 5992 or 8832 (on 32- / 64-bit platform)
```

expression

"expression"

expression(...)

is.expression(x)
as.expression(x, ...)

... expression
 as.expression

x

[callparse](#)

"expression"[[[
expressionis.expressionexpression

expression"expression"
is.expressionTRUEexprFALSE
as.expressionas.vector(type = "expression")[as.vector](#)NULL[as.symbol](#)[list](#)[typeof](#)

[calleva](#)[function](#)[text](#)[legend](#)[plot](#)[math](#)

```
length(ex1 <- expression(1 + 0:9)) # 1
ex1
eval(ex1) # 1:10

length(ex3 <- expression(u, 2, u + 0:9)) # 3
mode(ex3 [3]) # expression
mode(ex3[[3]]) # call
## but not all components are 'call's :
sapply(ex3, mode ) # name numeric call
sapply(ex3, typeof) # symbol double language
rm(ex3)
```

Extract

```
x[i]
x[i, j, ... , drop = TRUE]
x[[i, exact = TRUE]]
x[[i, j, ..., exact = TRUE]]
x$name
getElement(object, name)
```

```
x[i] <- value
x[i, j, ...] <- value
x[[i]] <- value
x$name <- value
```

```
xobject
ij...      numericcharacterNULLas.integertruncnamesdimnames
           [ij...ij...
           [ixi
           NULLinteger(0)
name       names
drop       TRUEdrop
exact      [[NAFALSE
value      x
```

```
[.data.frame[.factor[<-[<-[<-[<-
[[[[$[
x[[[]]
is.recursive$NULL
namesdimdimnames
xxvaluenamesdimdimnames
```

```
[[[names[c(abc = 123)][1]
ifactor[as.character(i)]
attributes
```



```
x
kk[
drop
NANANA
NANA""
x
```

```
[
[[[x$name[[name], exact = FALSE]][exact
getElement(x, name)x[[name, exact = TRUE]]slot(x, name)
[[[
[[ipalist[[i]]alist[[i1]]...[[ip]]
NULL
NULLNULLx[i] <- list(NULL)
$<-NULLxxlist()[[<-
```

```
[[get(i, env = x, inherits = FALSE)NULL$<-[[<-assign(i, value, env = x, inherits
= FALSE)x
```

```
NANANULL00
NANAvalue
```

```
m[j = 2, i = 1]m[2, 1]m[1, 2]
data.frame[.data.frame]ij
dropexact
```

```
x
$$<-namenname
```

```
pmatch[[exact
$options(warnPartialMatchDollar = TRUE)
""NA""
```

```
notSubsettableErrorobject
subscriptOutOfBoundsErrorobjectsubscriptindex
```

namespmatch
listarraymatrix
[.data.frame[.factor
Syntax
NULL

```
x <- 1:12
m <- matrix(1:6, nrow = 2, dimnames = list(c("a", "b"), LETTERS[1:3]))
li <- list(pi = pi, e = exp(1))
x[10]           # the tenth element of x
x <- x[-1]      # delete the 1st element of x
m[1,]           # the first row of matrix m
m[1, , drop = FALSE] # is a 1-row matrix
m[,c(TRUE,FALSE,TRUE)]# logical indexing
m[cbind(c(1,2,1),3:1)]# matrix numeric index
ci <- cbind(c("a", "b", "a"), c("A", "C", "B"))
m[ci]           # matrix character index
m <- m[,-1]      # delete the first column of m
li[[1]]         # the first element of list li
y <- list(1, 2, a = 4, 5)
y[c(3, 4)]      # a list containing elements 3 and 4 of y
y$a             # the element of y named a

## non-integer indices are truncated:
(i <- 3.999999999) # "4" is printed
(1:5)[i] # 3

## named atomic vectors, compare "[" and "[[" :
nx <- c(abc = 123, pi = pi)
nx[1] ; nx["pi"] # keeps names, whereas "[" does not:
nx[[1]] ; nx[["pi"]]
```

```
## recursive indexing into lists
z <- list(a = list(b = 9, c = "hello"), d = 1:5)
unlist(z)
z[[c(1, 2)]]
z[[c(1, 2, 1)]] # both "hello"
z[[c("a", "b")]] <- "new"
unlist(z)
```

```
## check $ and [[ for environments
e1 <- new.env()
e1$a <- 10
e1[["a"]]
e1[["b"]] <- 20
e1$b
ls(e1)
```

```
## partial matching - possibly with warning :
```

```

stopifnot(identical(li$p, pi))
op <- options(warnPartialMatchDollar = TRUE)
stopifnot( identical(li$p, pi), #-- a warning
  inherits(tryCatch (li$p, warning = identity), "warning"))
## revert the warning option:
options(op)

```

Extract.data.frame

```

## S3 method for class 'data.frame'
x[i, j, drop = ]
## S3 replacement method for class 'data.frame'
x[i, j] <- value
## S3 method for class 'data.frame'
x[ [..., exact = TRUE]]
## S3 replacement method for class 'data.frame'
x[[i, j]] <- value
## S3 replacement method for class 'data.frame'
x$name <- value

x
ij...      [[[numericcharacter[logicalas.integer[
name
drop        TRUE
value       NULL
exact       [

[[[x[i]x[[i]]drop
data.frame$x$nameExtract$value
[[[x[i, j]x[[i, j]][[xjxj[i][
[make.unique
drop = TRUE

[NULL
x[i]i[xx
[[[[[exact = FALSEexact = NAMatch

[NULL
[[NULL
$NULL
[<-[[<-$<-

```

```
[[[valuerep  
[  
[[[  
data.frameas.data.frame
```

```
drop = FALSEdrop = TRUE  
dropexact
```

```
subsetdata.frameExtract
```

```
sw <- swiss[1:5, 1:4] # select a manageable subset  
  
sw[1:3]      # select columns  
sw[, 1:3]    # same  
sw[4:5, 1:3] # select rows and columns  
sw[1]        # a one-column data frame  
sw[, 1, drop = FALSE] # the same  
sw[, 1]      # a (unnamed) vector  
sw[[1]]      # the same  
sw$Fert      # the same (possibly w/ warning, see ?Extract)  
  
sw[1,]       # a one-row data frame  
sw[1,, drop = TRUE] # a list  
  
sw["C", ] # partially matches  
sw[match("C", row.names(sw)), ] # no exact match  
try(sw[, "Ferti"]) # column names must match exactly  
  
sw[sw$Fertility > 90,] # logical indexing, see also ?subset  
sw[c(1, 1:2), ]       # duplicate row, unique row names are created  
  
sw[sw <= 6] <- 6 # logical matrix indexing  
sw  
  
## adding a column  
sw["new1"] <- LETTERS[1:5] # adds a character column  
sw[,"new2"] <- letters[1:5] # ditto  
sw[, "new3"] <- LETTERS[1:5] # ditto  
sw$new4 <- 1:5  
sapply(sw, class)  
sw$new # -> NULL: no unique partial match  
sw$new4 <- NULL # delete the column  
sw  
sw[6:8] <- list(letters[10:14], NULL, aa = 1:5)  
# update col. 6, delete 7, append  
sw
```

```

## matrices in a data frame
A <- data.frame(x = 1:3, y = I(matrix(4:9, 3, 2)),
               z = I(matrix(letters[1:9], 3, 3)))
A[1:3, "y"] # a matrix
A[1:3, "z"] # a matrix
A[, "y"]    # a matrix
stopifnot(identical(colnames(A), c("x", "y", "z")), ncol(A) == 3L,
          identical(A[, "y"], A[1:3, "y"]),
          inherits (A[, "y"], "AsIs"))

## keeping special attributes: use a class with a
## "as.data.frame" and "[" method;
## "avector" := vector that keeps attributes.  Could provide a constructor
## avector <- function(x) { class(x) <- c("avector", class(x)); x }
as.data.frame.avector <- as.data.frame.vector

`[,.avector` <- function(x,i,...) {
  r <- NextMethod("[")
  mostattributes(r) <- attributes(x)
  r
}

d <- data.frame(i = 0:7, f = gl(2,4),
               u = structure(11:18, unit = "kg", class = "avector"))
str(d[2:4, -1]) # 'u' keeps its "unit"

```

Extract.factor

```

## S3 method for class 'factor'
x[... , drop = FALSE]
## S3 method for class 'factor'
x[[...]]
## S3 replacement method for class 'factor'
x[...] <- value
## S3 replacement method for class 'factor'
x[[...]] <- value

```

```

x
...           Extract
drop
value

```

```

valuelevels(x)
contrastsdrop = TRUE
[[exact

```

```
xdrop = TRUE
```

factorExtract

```
## following example(factor)
(ff <- factor(substring("statistics", 1:10, 1:10), levels = letters))
ff[, drop = TRUE]
factor(letters[7:10])[2:3, drop = TRUE]
```

Extremes

```
pmax*()pmin*()
```

```
max(..., na.rm = FALSE)
min(..., na.rm = FALSE)
```

```
pmax(..., na.rm = FALSE)
pmin(..., na.rm = FALSE)
```

```
pmax.int(..., na.rm = FALSE)
pmin.int(..., na.rm = FALSE)
```

```
...
na.rm          TRUEFALSE
```

```
maxminintegerlogicalintegerdouble
na.rmFALSENANANA
+Inf-Infmin(x1, min(x2)) == min(x1, x2)xmax(x) == -Infmin(x) == +Inflength(x) ==
0pmaxpminNANAna.rm = TRUE
pmaxpminattributesnamesdimS4
pmax.intpmin.int
maxminSummary...
NaNNaNANANaNAmax(NA, Inf) == NAInf
NA""""
```

```
minmaxpminpmax
```

```
minmaxNAInf-Inf
```

```
maxminSummaryx, ..., na.rm
```

```
NULLinteger(0)
```

```
pmaxpminis.narep
```

```
rangewhich.minwhich.max
```

```
min
```

```
require(stats); require(graphics)
```

```
min(5:1, pi) #-> one number
```

```
pmin(5:1, pi) #-> 5 numbers
```

```
x <- sort(rnorm(100)); cH <- 1.35
```

```
pmin(cH, quantile(x)) # no names
```

```
pmin(quantile(x), cH) # has names
```

```
plot(x, pmin(cH, pmax(-cH, x)), type = "b", main = "Huber's function")
```

```
cut01 <- function(x) pmax(pmin(x, 1), 0)
```

```
curve(x^2 - 1/4, -1.4, 1.5, col = 2)
```

```
curve(cut01(x^2 - 1/4), col = "blue", add = TRUE, n = 500)
```

```
## pmax(), pmin() preserve attributes of *first* argument
```

```
D <- diag(x = (3:1)/4) ; n0 <- numeric()
```

```
stopifnot(identical(D, cut01(D)),
```

```
           identical(n0, cut01(n0)),
```

```
           identical(n0, cut01(NULL)),
```

```
           identical(n0, pmax(3:1, n0, 2)),
```

```
           identical(n0, pmax(n0, 4)))
```

```
extSoftVersion
```

```
extSoftVersion()
```

zlib	zlib
bzlib	bzlibbzip2
xz	liblzmaxz
libdeflate	libdeflate""
PCRE	PCRE
ICU	ICU""
TRE	libtre
iconv	iconv
readline	readline""libediteditline>EditLine wrapper"readline
BLAS	BLAS""

bzlibpcretre
 iconv"GNU libiconv 1.14""glibc 2.18""win_iconv"
 BLASlibR.solibR.dylibRlibRblas.solibRblas.dyliblibRblas.solibRblas.dylib

```

libcurlVersionlibCurl
La_version
La_library
grSoftVersion
tclVersion
pcre_config

```

```

extSoftVersion()
## the PCRE version
sub(".*", "", extSoftVersion()["PCRE"])

```

factor

```

factororderedTRUEordered
is.factoris.orderedas.factoras.ordered

```



```
factor(x = character(), levels, labels = levels,  
       exclude = NA, ordered = is.ordered(x), nmax = NA)
```

```
ordered(x = character(), ...)
```

```
is.factor(x)  
is.ordered(x)
```

```
as.factor(x)  
as.ordered(x)
```

```
addNA(x, ifany = FALSE)
```

```
.valid.factor(object)
```

```
x  
levels      xas.character(x)xsort(unique(x))  
labels      levelsexcludelabelsx  
exclude     xcharacter  
ordered  
nmax  
...         ordered(.)ordered  
ifany       NAany(is.na(x))  
object
```

```
xas.characterorder  
options("contrasts")  
excludelabelsx[i]levels[j]ix[i]levelsiNA  
excludelabels  
factor(x, exclude = NULL)NAexcludeexcludecharacterxexcludexexclude  
NAxexclude = NULLNA<NA>  
NAis.naais.na(f)[i] <- TRUEis.na<NA>NA  
is.factor  
levelsuniquexnmax  
cunlist
```

```
factor"factor"x"levels"character!anyDuplicated(.)orderedordered()c("ordered",  
"factor")factor(x)attributes(x)"names""levels""class"
```

```
factor[.factor
```

```
is.factorTRUEFALSEis.orderedTRUEFALSE
```

```
as.factorfactor
```

```
as.ordered(x)xordered(x)
```

```
addNANANA
```

```
.valid.factor(object)levels(object)TRUEvalidObject(<factor>)
```

```
"levels"as.numericfas.numeric(levels(f))[f]as.numeric(as.character(f))
```

NA

```
"factor""ordered"OpsminmaxrangeSummary"ordered"Math  
==!=
```

```
[.factor  
glClevelsnlevelsunclass
```

```
(ff <- factor(substring("statistics", 1:10, 1:10), levels = letters))  
as.integer(ff)      # the internal codes  
(f. <- factor(ff))  # drops the levels that do not occur  
ff[, drop = TRUE]   # the same, more transparently
```

```
factor(letters[1:20], labels = "letter")
```

```
class(ordered(4:1)) # "ordered", inheriting from "factor"  
z <- factor(LETTERS[3:1], ordered = TRUE)  
## and "relational" methods work:  
stopifnot(sort(z)[c(1,3)] == range(z), min(z) < max(z))
```

```
## suppose you want "NA" as a level, and to allow missing values.  
(x <- factor(c(1, 2, NA), exclude = NULL))  
is.na(x)[2] <- TRUE  
x # [1] 1    <NA> <NA>  
is.na(x)  
# [1] FALSE  TRUE  FALSE
```

```
## More rational, since R 3.4.0 :  
factor(c(1:2, NA), exclude = "" ) # keeps <NA> , as  
factor(c(1:2, NA), exclude = NULL) # always did  
## exclude = <character>  
z # ordered levels 'A < B < C'  
factor(z, exclude = "C") # does exclude  
factor(z, exclude = "B") # ditto
```

```
## Now, labels maybe duplicated:  
## factor() with duplicated labels allowing to "merge levels"  
x <- c("Man", "Male", "Man", "Lady", "Female")
```

```
## Map from 4 different values to only two levels:
(xf <- factor(x, levels = c("Male", "Man", "Lady", "Female"),
              labels = c("Male", "Male", "Female", "Female")))
#> [1] Male   Male   Male   Female Female
#> Levels: Male Female

## Using addNA()
Month <- airquality$Month
table(addNA(Month))
table(addNA(Month, ifany = TRUE))
```

file.access

```
file.access(names, mode = 0)
```

```
names      path.expand
mode
```

```
modexor0:7
```

```
file.access()try
```

```
0-1
```

```
access
```

```
file.infoSys.chmodtry
file_test
```

```
fa <- file.access(dir("."))
```

```
table(fa) # count successes & failures
```

`file.choose`

`file.choose(new = FALSE)`

`new`

[`list.files`](#)

`file.info`

`file.info(..., extra_cols = TRUE)`

`file.mode(...)`
`file.mtime(...)`
`file.size(...)`

`...` [`path.expand`](#)
`extra_cols`

[`file.exists`](#)

`.exe.com.cmd.bat` [`file.access`](#)

```

file.info()

size
isdir
mode          "octmode"644
mtimectimeatime
              "POSIXct"

      uid
      gidNA
      "no""msdos""win16""win32""win64""unknown"

      NA

extra_cols
NA
uidgidunamegrnameuname
ctime
"POSIXct"strftime
file.mode()file.mtime()file.size()

stat

```

[Sys.readlinkfilesfile.accesslist.filesDateTimeClasses](#)
[Sys.chmod](#)

```

ncol(finf <- file.info(dir())) # at least six
finf # the whole list
## Those that are more than 100 days old :
finf <- file.info(dir(), extra_cols = FALSE)
finf[difftime(Sys.time(), finf[, "mtime"], units = "days") > 100 , 1:4]

file.info("no-such-file-exists")

## E.g., for R-core, in a R-devel version:
if(Sys.info()[["sysname"]] == "Linux")
  sort(file.mtime(file.path(R.home("bin"),
                           c("",
                              file.path(c("", "exec"), "R"))))
        ))

```

`file.path`

```
file.path(..., fsep = .Platform$file.sep)
```

```
...
```

```
fsep
```

```
paste
```

```
PATHR_LIBSfsep = .Platform$path.sep
```

```
/d://\
```

```
fsep
```

```
Encoding
```

```
/\
```

```
basenamenormalizePathpath.expand
```

`file.show`

```
file.show(..., header = rep("", nfiles),
           title = "R Information",
           delete.file = FALSE, pager = getOption("pager"),
           encoding = "")
```

```
...
```

```
header      ...
```

```
title       title
```

```
delete.file
```

```
pager
```

```
encoding
```

[page](#)

```
pagerPATHR_HOME/bin/pagerPAGERlessmorepager
"internal""console"pager(files, header, title, delete.file)file.show
R.apppager
delete.file
```

[file.exists](#)[list.files](#)
[helpRShowDoc](#)[file.show](#)
[getOption\("pdfviewer"\)](#)[system](#)
[file.edit](#)

```
file.show(file.path(R.home("doc"), "COPYRIGHTS"))
```

files

```
file.create(..., showWarnings = TRUE)
file.exists(...)
file.remove(...)
file.rename(from, to)
file.append(file1, file2)
file.copy(from, to, overwrite = recursive, recursive = FALSE,
          copy.mode = TRUE, copy.date = FALSE)
file.symlink(from, to)
file.link(from, to)
```

```
...file1file2
fromto      file.copyfile.symlinkto
overwrite
showWarnings
recursive    tofromcp -R
copy.mode
copy.date    Sys.setFileTime
```

```

...path.expandfile.exists
file.create
file.existsstatstatfile.accessfile.accessexists
file.remove
file.renamefromtotoman 2 renamefile.renamefromtoto
file.append
file.copyfile.appendoverwrite = TRUEetocopy.mode = TRUEcopy.date = TRUE
file.symlinkfile.linkfile.symlinkto
Encoding

```

```

showWarnings = TRUEfile.create

```

```

file.renamefile.rename

```

```

file.infofile.accessfile.pathfile.showlist.filesunlinkbasenamepath.expand
dir.create
Sys.glob
file_testSys.readlink
https://en.wikipedia.org/wiki/Hard\_linkhttps://en.wikipedia.org/wiki/Symbolic\_
link

```

```

cat("file A\n", file = "A")
cat("file B\n", file = "B")
file.append("A", "B")
file.create("A") # (trashing previous)
file.append("A", rep("B", 10))
if(interactive()) file.show("A") # -> the 10 lines from 'B'
file.copy("A", "C")
dir.create("tmp")
file.copy(c("A", "B"), "tmp")
list.files("tmp") # -> "A" and "B"
setwd("tmp")
file.remove("A") # the tmp/A file
file.symlink(file.path("../", c("A", "B")), ".")
# |--> (TRUE,FALSE) : ok for A but not B as it exists already
setwd("../")
unlink("tmp", recursive = TRUE)
file.remove("A", "B", "C")

```

files2

```
dir.exists(paths)
dir.create(path, showWarnings = TRUE, recursive = FALSE, mode = "0777")
Sys.chmod(paths, mode = "0777", use_umask = TRUE)
Sys.umask(mode = NA)
```

```
path          path.expand
paths         path.expand
showWarnings
recursive     mkdir -p
mode          as.octmodeSys.chmodpaths
use_umask     umask
```

```
dir.existsfile.exists
dir.createrecursive = TRUEumaskmkdirmkdirman 2 mkdir
dir.create("G.S.")"G.S"
Sys.chmoddir.createchmodman 2 chmod
Sys.umaskumaskmode = NA"0"umaskman 2 umask
```

```
dir.existsTRUEFALSE
dir.createSys.chmoddir.createshowWarnings = TRUEdir.createrecursive = TRUE
Sys.umaskumask"octmode"modeNA
file.existspathpaths
```

```
file.infofile.existsfile.pathlist.filesunlinkbasenamepath.expand
```

```
## Not run:
## Fix up maximal allowed permissions in a file tree
Sys.chmod(list.dirs("."), "777")
f <- list.files(".", all.files = TRUE, full.names = TRUE, recursive = TRUE)
Sys.chmod(f, (file.mode(f) | "664"))

## End(Not run)
```

find.package

```
find.package(package, lib.loc = NULL, quiet = FALSE,
             verbose = getOption("verbose"))
```

```
path.package(package, quiet = FALSE)
```

```
packageNotFoundError(package, lib.loc, call = NULL)
```

```
package
```

```
lib.loc      NULLNULL.libPaths()
```

```
quiet
```

```
verbose      TRUE
```

```
call
```

```
find.packagelib.locNULLquiet = TRUEverboseMetaDESCRIPTIONversionlib.loc
```

```
find.packagerequire
```

```
path.packagequiet = TRUE
```

```
packageNotFoundErrorpackageNotFoundErrorpackagelib.loc
```

```
path.expandnormalizePath
```

```
try(find.package("knitr"))
```

```
## will not give an error, maybe a warning about *all* locations it is found:
```

```
find.package("kitty", quiet=TRUE, verbose=TRUE)
```

```
## Find all .libPaths() entries a package is found:
```

```
findPkgAll <- function(pkg)
```

```
  unlist(lapply(.libPaths(), function(lib)
```

```
    find.package(pkg, lib, quiet=TRUE, verbose=FALSE)))
```

```
findPkgAll("MASS")
```

```
findPkgAll("knitr")
```

findInterval

```
vecxi <- findInterval(x,v)jxv_{i_j} \le x_j < v_{i_j+1}v_0 := -\infty v_{N+1} := +\infty N <- length(v)
rightmost.closedall.inside
```

```
findInterval(x, vec, rightmost.closed = FALSE, all.inside = FALSE,
             left.open = FALSE, checkSorted = TRUE, checkNA = TRUE)
```

```
x
vec          N
rightmost.closed
              vec[N-1] .. vec[N]
all.inside    1,...,N-101NN-1
left.open     ≤<>≥rightmost.closed
checkSorted   vecis.unsorted(vec)FALSEvec
checkNA       x[i]is.na(.)FALSENAx[]
```

```
findIntervalxvecapply( outer(x, vec, `>=`), 1, sum)O(n \log N)n <- length(x)N <-
length(vec)xO(n)
```

```
findInterval(t, sort(X))nF_n(t; X_1, \dots, X_n)F_nX_1, \dots, X_n
```

```
rightmost.closed = TRUEEx[j] = vec[N]= \max vecN - 1
```

```
left.open = TRUE
```

```
identical(
  findInterval( x,  v,      left.open= TRUE, ...) ,
  N - findInterval(-x, -v[N:1], left.open=FALSE, ...) )
```

```
N <- length(vec)
```

```
length(x)0:NNAN <- length(vec)1:(N-1)all.inside = TRUENAxInfxvec
```

```
approx(*, method = "constant")findInterval()ecdfnfindInterval(.)
```

```

x <- 2:18
v <- c(5, 10, 15) # create two bins [5,10) and [10,15)
cbind(x, findInterval(x, v))

N <- 100
X <- sort(round(stats::rt(N, df = 2), 2))
tt <- c(-100, seq(-2, 2, length.out = 201), +100)
it <- findInterval(tt, X)
tt[it < 1 | it >= N] # only first and last are outside range(X)
stopifnot(identical(it, ## suppressing the checks is faster *BUT* dangerous, unless
                      ## you *know* that X is sorted and tt contains no NA's
                      findInterval(tt, X, checkSorted=FALSE, checkNA=FALSE)))

## 'left.open = TRUE' means "mirroring" :
N <- length(v)
stopifnot(identical(
  findInterval( x, v, left.open=TRUE) ,
  N - findInterval(-x, -v[N:1])))

```

force

force(x)

x

force

force

```

f <- function(y) function() y
lf <- vector("list", 5)
for (i in seq_along(lf)) lf[[i]] <- f(i)
lf[[1]]() # returns 5

g <- function(y) { force(y); function() y }
lg <- vector("list", 5)
for (i in seq_along(lg)) lg[[i]] <- g(i)
lg[[1]]() # returns 1

## This is identical to
g <- function(y) { y; function() y }

```

forceAndCall

forceAndCall(n, FUN, ...)

n
FUN
... FUN

forceAndCallFUN...FUNnFUNFUN(...)
forceAndCall[apply](#)

[forcepromiseclosure](#)

Foreign

.C(.NAME, ..., NAOK = FALSE, DUP = TRUE, PACKAGE, ENCODING)
.Fortran(.NAME, ..., NAOK = FALSE, DUP = TRUE, PACKAGE, ENCODING)

.NAME "[NativeSymbolInfo](#)" "[RegisteredNativeSymbol](#)" "[NativeSymbol](#)"
...
NAOK TRUE[NANaNI](#)[nf](#) FALSE [NANaNI](#)[nf](#)
PACKAGE .NAME.so.dll

DUPENCODING

[.Call.External](#)
.NAME.....NAME

...

integer	int *	integer
numeric	double *	double precision
	float *	real
complex	Rcomplex *	double complex
logical	int *	integer
character	char **	
raw	unsigned char *	
list	SEXP *	
	SEXP	

```
integer logical int long int long
logical integer logical
double *double precision Csingle TRUEas.singlesingle
RcomplexComplex.htypedef struct {double r; double i;}double complex
0FALSE1TRUEINT_MIN = -2147483648NANAOK = TRUEINT_MINTRUE
NA.Cchar"NA".Call
.Fortrancharacter*255
.C.Call

.Fortran
.Fortran.Fortran
.Fortran.Ciso_c_binding
```

```
options(CBoundsCheck = TRUE).C
```

```
PACKAGE.NAME
PACKAGE = "base"
```

```
dyn.load.Call
```

formals

function

```
formals(fun = sys.function(sys.parent()), envir = parent.frame())
formals(fun, envir = environment(fun)) <- value
```

```
fun          function
envir        environmentget()fun
value        listpairlistNULL
```

```
funenvirformals
```

```
formals(args(f))f
```

```
formalspairlistNULL
```

```
attributes
```

```
formalArgsnames(formals(.))args
alistvalue
functionformalsbodyenvironment
```

```
require(stats)
formals(lm)
```

```
## If you just want the names of the arguments, use formalArgs instead.
names(formals(lm))
methods:: formalArgs(lm)    # same
```

```
## formals returns a pairlist. Arguments with no default have type symbol (aka name).
str(formals(lm))
```

```
## formals returns NULL for primitive functions. Use it in combination with
## args for this case.
is.primitive(`+`)
formals(`+`)
formals(args(`+`))
```

```
## You can overwrite the formal arguments of a function (though this is
## advanced, dangerous coding).
f <- function(x) a + b
formals(f) <- alist(a = , b = 3)
f      # function(a, b = 3) a + b
f(2)   # result = 5
```

format

```
format(x, ...)
```

```
## Default S3 method:
```

```
format(x, trim = FALSE, digits = NULL, nsmall = 0L,  
       justify = c("left", "right", "centre", "none"),  
       width = NULL, na.encode = TRUE, scientific = NA,  
       big.mark = "", big.interval = 3L,  
       small.mark = "", small.interval = 5L,  
       decimal.mark = getOption("OutDec"),  
       zero.print = NULL, drop0trailing = FALSE, ...)
```

```
## S3 method for class 'data.frame'
```

```
format(x, ..., justify = "none", cut.names = TRUE)
```

```
## S3 method for class 'factor'
```

```
format(x, ...)
```

```
## S3 method for class 'AsIs'
```

```
format(x, width = 12, ...)
```

```
x
```

```
trim          FALSETRUE
```

```
digits        xNULLgetOption("digits")nsmallsignif
```

```
nsmall        0 <= nsmall <= 20
```

```
justify
```

```
cut.names      logicallistas.data.frame()
```

```
width          defaultNULL0
```

```
AsIsNULL12
```

```
na.encode      NANA"NA"
```

```
scientific     options("scipen")
```

```
...
```

```
big.markbig.intervalsmall.marksmall.intervaldecimal.markzero.printdrop0trailing  
prettyNum
```

```
formatformat.Dateformat.POSIXctformat.octmodeformat.dist
```

```
format.data.frameformatas.character"AsIs"
```

```
format.factorjustify
```

```
format.AsIswidthtoString
```



```

ncharprint.defaultprint(quote = FALSE)catna.encode = FALSE
digitsbig.*small.*print.defaultdigits >= 16
as.character
format.default(x)isS4(x)
getOption("OutDec")decimal.markscientificgetOption("scipen")

xx
x
xformat.default(x, ...)unlistpaste(collapse = ", ")trim = TRUE, justify = "none"

```

```

format.info
formatCpasteas.charactersprintfprintprettyNumtoStringencodeString

```

```

format(1:10)
format(1:10, trim = TRUE)

zz <- data.frame("(row names)"= c("aaaaa", "b"), check.names = FALSE)
format(zz)
format(zz, justify = "left")

## use of nsmall
format(13.7)
format(13.7, nsmall = 3)
format(c(6.0, 13.1), digits = 2)
format(c(6.0, 13.1), digits = 2, nsmall = 1)

## use of scientific
format(2^31-1)
format(2^31-1, scientific = TRUE)
## scientific = numeric scipen (= {sci}entific notation {pen}alty) :
x <- c(1e5, 1000, 10, 0.1, .001, .123)
t(sapply(setNames(,-4:1),
          \(sci) sapply(x, format, scientific=sci)))

## a list
z <- list(a = letters[1:3], b = (-pi+0i)^((-2:2)/2), c = c(1,10,100,1000),
          d = c("a", "longer", "character", "string"),
          q = quote( a + b ), e = expression(1+x))
## can you find the "2" small differences?
(f1 <- format(z, digits = 2))
(f2 <- format(z, digits = 2, justify = "left", trim = FALSE))
f1 == f2 ## 2 FALSE, 4 TRUE

## A "minimal" format() for S4 objects without their own format() method:
cc <- methods::getClassDef("standardGeneric")
format(cc) ## "<S4 class .....>"

```

format.info

`format(x, digits, nsmall)`

`format.info(x, digits = NULL, nsmall = 0)`

x	<code>format(x, ...)</code>
digits	<code>xNULLgetOption("digits")</code>
nsmall	<code>format(..., nsmall)</code>

`integervectorr`

`formatwidth = NULL`

r[1]	<code>format(x)</code>
r[2]	
r[3]	<code>0:2 ≥ 1r[3]+1</code>

`formatdigits >= 16formatC`

```
dd <- options("digits") ; options(digits = 7) #-- for the following
format.info(123)  # 3 0 0
format.info(pi)   # 8 6 0
format.info(1e8)  # 5 0 1 - exponential "1e+08"
format.info(1e222) # 6 0 2 - exponential "1e+222"
```

```
x <- pi*10^c(-10,-2,0:2,8,20)
names(x) <- formatC(x, width = 1, digits = 3, format = "g")
cbind(sapply(x, format))
t(sapply(x, format.info))
```

```
## using at least 8 digits right of "."
t(sapply(x, format.info, nsmall = 8))
```

```
# Reset old options:
options(dd)
```

`format.pval`

`format.pval`

```
format.pval(pv, digits = max(1, getOption("digits") - 2),
            eps = .Machine$double.eps, na.form = "NA", ...)
```

```
pv
digits
eps
na.form      NA
...          formatnsmall
```

```
format.pval(print.summary.lmeps"< [eps]"format(eps, digits)
```

```
format.pval(c(stats::runif(5), pi^-100, NA))
format.pval(c(0.1, 0.0001, 1e-27))
```

`formatC`

```
formatC()C
prettyNum()format.default
.format.zeros(x)prettyNum()x
```

```
formatC(x, digits = NULL, width = NULL,
        format = NULL, flag = "", mode = NULL,
        big.mark = "", big.interval = 3L,
        small.mark = "", small.interval = 5L,
        decimal.mark = getOption("OutDec"),
        preserve.width = "individual",
        zero.print = NULL, replace.zero = TRUE,
        drop0trailing = FALSE)
```

```
prettyNum(x, big.mark = "", big.interval = 3L,
          small.mark = "", small.interval = 5L,
```

```

        decimal.mark = getOption("OutDec"), input.d.mark = decimal.mark,
        preserve.width = c("common", "individual", "none"),
        zero.print = NULL, replace.zero = FALSE,
        drop0trailing = FALSE, is.cmplx = NA,
        ...)

.format.zeros(x, zero.print, nx = suppressWarnings(as.numeric(x)),
              replace = FALSE, warn.non.fitting = TRUE)

x          complexprettyNum()
digits      format = "f"format = "g"= "e"= "fg"
            format = "f"
width       digitwidthwidthdigits + 1width = 0width = digitwidth < 0flag =
            "-"width
format      "d""f""e""E""g""G""fg""s""d""g"
            "f"xxx.xxx"e""E"n.ddde+nnn.dddE+nn"g""G"x[i]flag#"g", "G"
            "fg""f"digitssignifdigitsflag#"
flag        formatC
            "0"
            "_"
            "+" "+"
            " " " "
            "#" format
            ","
            "I" glibc
            character
mode         "double""real""integer""character"x
big.mark     big.intervalbig
big.interval big.mark
small.mark   small.intervalsmall
small.interval small.mark
decimal.mark
input.d.mark xcharacterx
preserve.width big.marksmall.mark"common"format"individual"formatC()
zero.print   NULL
replace.zero replace
            zero.printxzero.printzero.print"      "replace[.zero]zero.print
            warn.non.fitting
            prettyNum().format.zeros(*, replace=replace.zero)
warn.non.fitting
            replace[.zero]zero.printwarning
drop0trailing "0""e+00"prettyNum()
is.cmplx      x"character"complexNAx
...           format
nx            xx

```

```

formatC(prettyNum()).format.zeros(*,          replace=replace.zero)zero.printNULL
drop0trailingbig.marksmall.markdecimal.mark
formatmodeformatC(123.45, mode = "double", format = "d")123
n.ddde+nennn.dddenenn.ddde+nn
formatCformatC(c(6.11, 13.1), digits = 2, format = "fg")c("6.1", " 13")format
prettyNumxxformat(<complex>)xformat(x[i],      ...)input.d.markprettyNum(x)x
characterformat(<number>)
gsubbig.marksmall.mark"\\\\"
formatC-0-0.000
big.markdecimal.mark

```

```

x
formatxsprintf(...)str_signif
formatCx

```

```

decimal.markformatC()printdecimal.mark = getOption("OutDec")

```

```

formatC

```

```

format
sprintf

```

```

xx <- pi * 10^(-5:4)
cbind(format(xx, digits = 4), formatC(xx))
cbind(formatC(xx, width = 9, flag = "-"))
cbind(formatC(xx, digits = 5, width = 8, format = "f", flag = "0"))
cbind(format(xx, digits = 4), formatC(xx, digits = 4, format = "fg"))

f <- (-2:4); f <- f*16^f
# Default ("g") format:
formatC(pi*f)
# Fixed ("f") format, more than one flag ('width' partly "enlarged"):
cbind(formatC(pi*f, digits = 3, width=9, format = "f", flag = "0+"))

formatC(      c("a", "Abc", "no way"), width = -7) # <=> flag = "-"
formatC(c((-1:1)/0,c(1,100)*pi), width = 8, digits = 1)

## note that some of the results here depend on the implementation

```

```

## of long-double arithmetic, which is platform-specific.
xx <- c(1e-12,-3.98765e-10,1.45645e-69,1e-70,pi*1e37,3.44e4)
##      1      2      3      4      5      6
formatC(xx)
formatC(xx, format = "fg")      # special "fixed" format.
formatC(xx[1:4], format = "f", digits = 75) #>> even longer strings

formatC(c(3.24, 2.3e-6), format = "f", digits = 11)
formatC(c(3.24, 2.3e-6), format = "f", digits = 11, drop0trailing = TRUE)

r <- c("76491283764.97430", "29.12345678901", "-7.1234", "-100.1", "1123")
## American:
prettyNum(r, big.mark = ",")
## Some Europeans:
prettyNum(r, big.mark = "'", decimal.mark = ",")

(dd <- sapply(1:10, function(i) paste((9:0)[1:i], collapse = "")))
prettyNum(dd, big.mark = "'")

## examples of 'small.mark'
pN <- stats::pnorm(1:7, lower.tail = FALSE)
cbind(format (pN, small.mark = " ", digits = 15))
cbind(formatC(pN, small.mark = " ", digits = 17, format = "f"))

cbind(ff <- format(1.2345 + 10^(0:5), width = 11, big.mark = "'"))
## all with same width (one more than the specified minimum)

## individual formatting to common width:
fc <- formatC(1.234 + 10^(0:8), format = "fg", width = 11, big.mark = "'")
cbind(fc)
## Powers of two, stored exactly, formatted individually:
pow.2 <- formatC(2^-(1:32), digits = 24, width = 1, format = "fg")
## nicely printed (the last line showing 5^32 exactly):
noquote(cbind(pow.2))

## complex numbers:
r <- 10.0000001; rv <- (r/10)^(1:10)
(zv <- (rv + 1i*rv))
op <- options(digits = 7) ## (system default)
(pnv <- prettyNum(zv))
stopifnot(pnv == "1+1i", pnv == format(zv),
          pnv == prettyNum(zv, drop0trailing = TRUE))
## more digits change the picture:
options(digits = 8)
head(fv <- format(zv), 3)
prettyNum(fv)
prettyNum(fv, drop0trailing = TRUE) # a bit nicer
options(op)

## The ' flag :
doLC <- FALSE # <= R warns, so change to TRUE manually if you want see the effect
if(doLC) {
  oldLC <- Sys.getlocale("LC_NUMERIC")
  Sys.setlocale("LC_NUMERIC", "de_CH.UTF-8")
}
formatC(1.234 + 10^(0:4), format = "fg", width = 11, flag = "'")
## --> ..... "      1'001" "      10'001" on supported platforms

```

```
if(doLC) ## revert, typically to "C" :  
  Sys.setlocale("LC_NUMERIC", oldLC)
```

formatDL

```
formatDL(x, y, style = c("table", "list"),  
         width = 0.9 * getOption("width"), indent = NULL)
```

```
x  
y          xx  
style      "table"@table"list"  
width  
indent     width/2width/3width/9
```

```
indent - 3
```

```
## Provide a nice summary of the numerical characteristics of the  
## machine R is running on:  
writeLines(formatDL(unlist(.Machine)))  
## Inspect Sys.getenv() results in "list" style (by default, these are  
## printed in "table" style):  
writeLines(formatDL(Sys.getenv(), style = "list"))
```

function

```
function( arglist ) expr  
\( arglist ) expr  
return(value)
```

```
arglist      namename = expression...  
expr  
value
```

```
valueNULLreturnon.exit
return
\(\x) x + 1function(x) x + 1
```

```
formalsbodyexprenvironment
```

```
args
formalsbodyenvironment
debuginvisiblereturn(.)
```

```
norm <- function(x) sqrt(x%*%x)
norm(1:4)

## An anonymous function:
(function(x, y){ z <- x^2 + y^2; x+y+z })(0:7, 1)
```

funprog

```
Reduce
Filter
FindPosition
Map
Negate
```

```
Reduce(f, x, init, right = FALSE, accumulate = FALSE, simplify = TRUE)
Filter(f, x)
Find(f, x, right = FALSE, nomatch = NULL)
Map(f, ...)
Negate(f)
Position(f, x, right = FALSE, nomatch = NA_integer_)
```



```

f          ReduceFilterFindPositionkMapkNegate
x
init      x
right
accumulate
simplify
nomatch
...      Map()mapplyMoreArgs

```

$$\text{initReduce} x v n > 1 \text{Reduce} f v l_1 = f(v_1, v_2) l_2 = f(l_1, v_3) l_{n-1} = f(l_{n-2}, v_n) r_{n-1} = f(v_{n-1}, v_n) r_{n-2} = f(v_{n-2}, r_{n-1}) r_1 = f(v_1, r_2) v f(2/3)/4 = 1/62/(3/4) = 8/3 v \text{NULL} f$$

```

Reducerreduce
FilterfxNaNANAFilterfilterremove-if-not
FindPositionfind-ifposition-ifrightnomatch
Mapmapplymapcar
Negatecomplementff

```

```
clusterMapmcmapplyMap
```

```

## A general-purpose adder:
add <- function(x) Reduce(`+`, x)
add(list(1, 2, 3))
## Like sum(), but can also used for adding matrices etc., as it will
## use the appropriate '+' method in each reduction step.
## More generally, many generics meant to work on arbitrarily many
## arguments can be defined via reduction:
F00 <- function(...) Reduce(F002, list(...))
F002 <- function(x, y) UseMethod("F002")
## F00() methods can then be provided via F002() methods.

## A general-purpose cumulative adder:
cadd <- function(x) Reduce(`+`, x, accumulate = TRUE)
cadd(seq_len(7))

## A simple function to compute continued fractions:
cfrac <- function(x) Reduce(function(u, v) u + 1 / v, x, right = TRUE)
## Continued fraction approximation for pi:
cfrac(c(3, 7, 15, 1, 292))
## Continued fraction approximation for Euler's number (e):
cfrac(c(2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8))

## Map() now recycles similar to basic Ops:
Map(`+`, 1, 1 : 3) ; 1 + 1:3
Map(`+`, numeric(), 1 : 3) ; numeric() + 1:3

```

```

## Iterative function application:
Funcall <- function(f, ...) f(...)
## Compute log(exp(acos(cos(0))))
Reduce(Funcall, list(log, exp, acos, cos), 0, right = TRUE)
## n-fold iterate of a function, functional style:
Iterate <- function(f, n = 1)
  function(x) Reduce(Funcall, rep.int(list(f), n), x, right = TRUE)
## Continued fraction approximation to the golden ratio:
Iterate(function(x) 1 + 1 / x, 30)(1)
## which is the same as
cfrac(rep.int(1, 31))
## Computing square root approximations for x as fixed points of the
## function t |-> (t + x / t) / 2, as a function of the initial value:
asqrt <- function(x, n) Iterate(function(t) (t + x / t) / 2, n)
asqrt(2, 30)(10) # Starting from a positive value => +sqrt(2)
asqrt(2, 30)(-1) # Starting from a negative value => -sqrt(2)

## A list of all functions in the base environment:
funs <- Filter(is.function, sapply(ls(baseenv()), get, baseenv()))
## Functions in base with more than 10 arguments:
names(Filter(function(f) length(formals(f)) > 10, funs))
## Number of functions in base with a '...' argument:
length(Filter(function(f)
  any(names(formals(f)) %in% "..."),
  funs))

## Find all objects in the base environment which are *not* functions:
Filter(Negate(is.function), sapply(ls(baseenv()), get, baseenv()))

```

```
gc
```

```
gcgcinfoverbose = FALSEverbose = TRUE
```

```
gc(verbose = getOption("verbose"), reset = FALSE, full = TRUE)
gcinfo(verbose)
```

```

verbose      TRUE
reset        TRUE
full         TRUE

```

```
gcgcfull = TRUE
```

```
gc
```

```
"Vcells"
```

```
gcinfo(TRUE)
```

```

Garbage collection 12 = 10+0+2 (level 0) ...
6.4 Mbytes of cons cells used (58%)
2.0 Mbytes of vectors used (32%)

```

```
gc"Ncells""Vcells""used""gc trigger"
"Ncells""Vcells"NA
gc(reset = TRUE)
gcinfo
```

Memorygctorture

```
gc.time()
reg.finalizer
```

```
gc() #- do it now
gcinfo(TRUE) #-- in the future, show when R does it
##          vvvvv use larger to *show* something
x <- integer(100000); for(i in 1:18) x <- c(x, i)
gcinfo(verbose = FALSE) #-- don't show it anymore
```

```
gc(TRUE)
```

```
gc(reset = TRUE)
```

```
gc.time
```

```
gc.time(on = TRUE)
```

```
on          TRUE
```

```
NA
```

```
gcproc.time
```

```
gc.time()
```

gctorture

```
gctorture(on = TRUE)
gctorture2(step, wait = step, inhibit_release = FALSE)
```

```
on
step          stepstep = 0
wait
inhibit_release
```

```
gctorture(TRUE)gctorture2step
gctorture2inhibit_release
gctorture2R_GCTORTUREstepR_GCTORTURE_WAITwaitR_GCTORTURE_INHIBIT_RELEASE
inhibit_release
```

get

getmget

```
get(x, pos = -1, envir = as.environment(pos), mode = "any",
    inherits = TRUE)

mget(x, envir = as.environment(-1), mode = "any", ifnotfound,
     inherits = FALSE)

dynGet(x, ifnotfound = , minframe = 1L, inherits = FALSE)
```

```

x                get
                 mget

posenvir
mode
inherits
ifnotfound      mgetlist
                 dynGetstop()

minframe

possearchenvironmentsys.frame-1getenvir
xinheritsTRUExxenvironment
modemodetypeofmode"function""numeric""integer""double""symbol""name""language"
"call""("mode = "S4"mode = "object"
mgetmodeifnotfoundxifnotfound
dynGet()sys.frame()

getR_MissingArg"getMissingError"
mgetifnotfound

a <- get(nam)assign(nam, a)anam
inherits = TRUEget

existsget0
assignget()
getAnywheregetFromNamespace

get("%0%")

## test mget
e1 <- new.env()
mget(letters, e1, ifnotfound = as.list(LETTERS))

## very low-level: get()ing the "missing argument", e.g., inside browser()
ls.str(E <- environment((\ (m) \(){}))()) # m : <missing>
str(E$m) # (empty) symbol
ee <- tryCatch(get("m",E), error = function(e) e)
str(ee)
ee

```

```

stopifnot(exprs = {
  inherits(ee, "missingArgError") # and
  inherits(ee, "getMissingError")
##
  inherits(tryCatch(get0("m", E), error=identity), "getMissingError")
  is.symbol(E$m) # => valid argument to get(), and *also* gets 'missing arg':
  inherits(tryCatch(get ( E$m ) , error=identity), "getMissingError")
})

```

getDLLRegisteredRoutines

[.C.Call.Fortran.External](#)

```
getDLLRegisteredRoutines(dll, addNames = TRUE)
```

```

dll          DLLInfo.dll.soMyPackage/libs/MyPackage.soMyPackage
              DLLInfodyn.loadlibrary.dynamgetLoadedDLLsDLLInfo
              DLLInfo
addNames      TRUEFALSENativeSymbolInfo

```

```
print
```

```

"DLLRegisteredRoutines".C.Call.Fortran.External"NativeRoutineList"
"NativeSymbolInfo"

```

```

name
address      NULL
dll          DLLInfo
numParameters

```

```
<duncan@wald.ucdavis.edu>
```

https://www.r-project.org/doc/Rnews/Rnews_2001-3.pdf

[getLoadedDLLsgetNativeSymbolInfo](#)

```

dlls <- getLoadedDLLs()
getDLLRegisteredRoutines(dlls[["base"]])

```

```
getDLLRegisteredRoutines("stats")
```

`getLoadedDLLs`

`dyn.load``getLoadedDLLs()``"DLLInfoList"list"DLLInfo"``name``path``dynamicLookup``handle "DLLHandle"``DLLInfo$[[x[["name"]]]x[["handle"]]]``handle``<duncan@wald.ucdavis.edu>``getDLLRegisteredRoutinesgetNativeSymbolInfo``getLoadedDLLs()``utils::tail(getLoadedDLLs(), 2) # the last 2 loaded ones, still a DLLInfoList`

`getNativeSymbolInfo`

`.Call.C.Fortran.External``getNativeSymbolInfo(name, PACKAGE, unlist = TRUE,
 withRegistrationInfo = FALSE)`

```

name
PACKAGE      "base"
unlist       unlistTRUEnameNativeSymbolInfoFALSENativeSymbolInfonameTRUE
withRegistrationInfo
              TRUEFALSE

```

[.Call](#)

NativeSymbolInfonameNativeSymbolInfo

```

name          name
address       withRegistrationInfoFALSENativeSymbolwithRegistrationInfoTRUE
              RegisteredNativeSymbol.Call.C.Fortran.External
dll
              PACKAGE

```

numParameters

CRoutineCallRoutineFortranRoutineExternalRoutine

nameunlistTRUENativeSymbolInfo

NativeSymbolInfopackagedllNativeSymbolInfo[getDLLRegisteredRoutines\(\)](#)

[nls](#)

https://www.r-project.org/doc/Rnews/Rnews_2001-3.pdf

[getDLLRegisteredRoutinesis.loaded.C.Fortran.External.Calldyn.load](#)

gettext

`stop()``warning()``message()``bindtextdomain()`

`gettext(..., domain = NULL, trim = TRUE)`

`ngettext(n, msg1, msg2, domain = NULL)`

`bindtextdomain(domain, dirname = NULL)`

`Sys.setLanguage(lang, unset = "en")`

```
...
trim      gettext()trim = FALSE\n
domain    characterNULL
n
msg1      n = 1
msg2      n = 0, 2, 3, ...
dirname
lang      character
unset     Sys.getenv("LANGUAGE")
```

`domainNULLgettextngettextgettextngettextstop()warning()message()evalq() "R-pkg"`

`domain = NAgettextngettext`

`""gettextngettextdomain = ""domain = NA`

`Sys.getlocale()``LANGUAGESys.setLanguage()`

`"R-pkg""pkg"`

`gettexttrim`

`ngettextgettext%dsprintfmsg1n == 1msg2`

`bindtextdomainmanNULLdirnamedirname = NULLbindtextdomain(*,*)NULL`

`bindtextdomain(NULL)textdomain(textdomain(NULL))TRUE`

`Sys.setLanguage(lang)LANGUAGEbindtextdomain(NULL)`

`gettext...`

`ngettext`

`bindtextdomainNULL`

`Sys.setLanguage()LANGUAGEattr(*, "ok")logicallangmessage`

```
LANGUAGElibintlgettextmusl
```

```
stopwarninggettext
```

```
xgettext
```

```
bindtextdomain("R") # non-null if and only if NLS is enabled
```

```
for(n in 0:3)
  print(sprintf(ngettext(n, "%d variable has missing values",
                        "%d variables have missing values"),
                n))
```

```
## Not run: ## for translation, those strings should appear in R-pkg.pot as
```

```
msgid          "%d variable has missing values"
msgid_plural    "%d variables have missing values"
msgstr[0]       ""
msgstr[1]       ""
```

```
## End(Not run)
```

```
miss <- "One only" # this line, or the next for the ngettext() below
miss <- c("one", "or", "another")
cat(ngettext(length(miss), "variable", "variables"),
    paste(sQuote(miss), collapse = ", "),
    ngettext(length(miss), "contains", "contain"), "missing values\n")
```

```
## better for translators would be to use
cat(sprintf(ngettext(length(miss),
                    "variable %s contains missing values\n",
                    "variables %s contain missing values\n"),
    paste(sQuote(miss), collapse = ", ")))
```

```
thisLang <- Sys.getenv("LANGUAGE", unset = NA) # so we can reset it
if(is.na(thisLang) || !nzchar(thisLang)) thisLang <- "en" # "factory" default
enT <- "empty model supplied"
Sys.setenv(LANGUAGE = "de") # may not always 'work'
gettext(enT, domain="R-stats")# "leeres Modell angegeben" (if translation works)
tget <- function() gettext(enT)
tget() # not translated as fn tget() is not from "stats" pkg/namespace
evalq(function() gettext(enT), asNamespace("stats"))() # *is* translated
```

```
## Sys.setLanguage() -- typical usage --
Sys.setLanguage("en") -> oldSet # does set LANGUAGE env.var
errMsg <- function(expr) tryCatch(expr, error=conditionMessage)
(errMsg(1 + "2") -> err)
Sys.setLanguage("fr")
errMsg(1 + "2")
Sys.setLanguage("de")
errMsg(1 + "2")
## Usually, you would reset the language to "previous" via
Sys.setLanguage(oldSet)
```

```

## A show off of translations -- platform (font etc) dependent:
## The translation languages available for "base" R in this version of R:
if(capabilities("NLS")) withAutoprint({
  langs <- list.files(bindtextdomain("R"),
    pattern = "^[a-z]{2}(_[A-Z]{2}|@quot)?$",
    langs
  txts <- sapply(setNames(,langs),
    function(lang) { Sys.setLanguage(lang)
      gettext("incompatible dimensions", domain="R-stats") })
  cbind(txts)
  (nTrans <- length(unique(txts)))
  (not_translated <- names(txts[txts == txts[["en"]]]))
})

## Here, we reset to the *original* setting before the full example started:
if(nzchar(thisLang)) { ## reset to previous and check
  Sys.setLanguage(thisLang)
  stopifnot(identical(errMsg(1 + "2"), err))
} # else staying at 'de' ..

```

getwd

getwdsetwd(dir)dir

getwd()
setwd(dir)

dir

getwdNULL//
setwdgetwd

[list.files](#)
[normalizePath](#)

```

(WD <- getwd())
if (!is.null(WD)) setwd(WD)

```

gl

```
gl(n, k, length = n*k, labels = seq_len(n), ordered = FALSE)
```

n

k

length

labels

ordered

1nklength

gl

`factor()`

```
## First control, then treatment:
gl(2, 8, labels = c("Control", "Treat"))
## 20 alternating 1s and 2s
gl(2, 1, 20)
## alternating pairs of 1s and 2s
gl(2, 2, 20)
```

grep

grep
grepv
grepl
regexpr
gregexpr
regexec
gregexec
pattern
sub
gsub

```
grep(pattern, x, ignore.case = FALSE, perl = FALSE, value = FALSE,
      fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

```
grepv(pattern, x, ignore.case = FALSE, perl = FALSE, value = TRUE,
       fixed = FALSE, useBytes = FALSE, invert = FALSE)
```

```
grepl(pattern, x, ignore.case = FALSE, perl = FALSE,
       fixed = FALSE, useBytes = FALSE)
```

```

sub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

gsub(pattern, replacement, x, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

regexpr(pattern, text, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

gregexpr(pattern, text, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

regexec(pattern, text, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

gregexec(pattern, text, ignore.case = FALSE, perl = FALSE,
    fixed = FALSE, useBytes = FALSE)

```

pattern	fixed = TRUE	as.character	regexpr	gregexpr	regexec
xtext		as.character			
ignore.case	FALSE	TRUE			
perl					
value	FALSE	integer	grep	TRUE	
fixed	TRUE	pattern			
useBytes	TRUE				
invert	TRUE				
replacement	sub	subfixed = FALSE	"\1""\9"	pattern	perl = TRUE
			"\U""\L""\E"	NANA	

```

fixed = TRUE
perl = TRUE
fixed = FALSE, perl = FALSE

```

```

*subsubpatternsgsubreplacementpattern""
regexprgregexprregexecgregexecpatternNANANA
grepgrepvvalue
grepgreplpattern
useBytes = TRUEregexpr"bytes"Encoding
useBytes = TRUE
regexprgregexprperl = TRUE

perl = TRUE

```

```
grep(value = FALSE)xinvert = TRUE
grep(value = TRUE)x
greplx
subgsubxxuseBytes = FALSEuseBytes = FALSEfixed = TRUEenc2native"bytes"x"bytes"
"bytes"useBytes = TRUE"bytes""unknown""bytes"iconv
regexprtext-1"match.length"-1useBytes = TRUEuseBytesTRUE"capture.start"
"capture.length""capture.names"
gregexprtextregexpr
regextext-1pattern"match.length"-1regexpr
gregexecregextexttext
perl = TRUE
```

```
gsubgregexprpattern = "\b"perl = TRUE
```

```
perl = TRUEfixed = TRUE
```

```
3 ×
```

```
useBytes = TRUEfixed = TRUEuseBytes = FALSEuseBytes = TRUE
xtextpcre_configextSoftVersionoptionsPCRE_studyPCRE_use_JITtests/PCRE.R
R_PCRE_JIT_STACK_MAXSIZE1100064PCRE_limit_recursion
```

```
ignore.caseperlvaluefixeduseBytesinverttext
```

```
[[:digit:]][:xdigit:]
```

```
https://github.com/laurikari/tre
https://www.pcre.org
```

```
grep
```

```
regexp
regmatchesregexprgregexprregexec
glob2rx
agrep
charmatchpmatchmatchstartsWith
tolowertoupperchartr
apropos
grepRaw
PCRE_limit_recursionPCRE_studyPCRE_use_JIT
extSoftVersionpcre_config
```

```

grep("[a-z]", letters)

txt <- c("arm", "foot", "lefroo", "bafoobar")
if(length(i <- grep("foo", txt)))
  cat("'foo' appears at least once in\n\t", txt, "\n")
i # 2 and 4
txt[i]

## Double all 'a' or 'b's; "\" must be escaped, i.e., 'doubled'
gsub("[ab]", "\\1_\\1_", "abc and ABC")

txt <- c("The", "licenses", "for", "most", "software", "are",
  "designed", "to", "take", "away", "your", "freedom",
  "to", "share", "and", "change", "it.",
  "", "By", "contrast,", "the", "GNU", "General", "Public", "License",
  "is", "intended", "to", "guarantee", "your", "freedom", "to",
  "share", "and", "change", "free", "software", "--",
  "to", "make", "sure", "the", "software", "is",
  "free", "for", "all", "its", "users")
( i <- grep("[gu]", txt) ) # indices
stopifnot( txt[i] == grep("[gu]", txt, value = TRUE) )

## Note that for some implementations character ranges are
## locale-dependent (but not currently). Then [b-e] in locales such as
## en_US may include B as the collation order is aAbBcCdDe ...
(ot <- sub("[b-e]", ".", txt))
txt[ot != gsub("[b-e]", ".", txt)]#- gsub does "global" substitution
## In caseless matching, ranges include both cases:
a <- grep("[b-e]", txt, value = TRUE)
b <- grep("[b-e]", txt, ignore.case = TRUE, value = TRUE)
setdiff(b, a)

txt[gsub("g", "#", txt) !=
  gsub("g", "#", txt, ignore.case = TRUE)] # the "G" words

regexpr("en", txt)

grexexpr("e", txt)

## Using grepl() for filtering
## Find functions with argument names matching "warn":
findArgs <- function(env, pattern) {
  nms <- ls(envir = as.environment(env))
  nms <- nms[is.na(match(nms, c("F", "T")))] # <-- work around "checking hack"
  aa <- sapply(nms, function(.) { o <- get(.)
    if(is.function(o)) names(formals(o)) })
  iw <- sapply(aa, function(a) any(grepl(pattern, a, ignore.case=TRUE)))
  aa[iw]
}
findArgs("package:base", "warn")

## trim trailing white space
str <- "Now is the time      "
sub(" +$", "", str) ## spaces only
## what is considered 'white space' depends on the locale.
sub("[[:space:]]+$", "", str) ## white space, POSIX-style

```

```

## what PCRE considered white space changed in version 8.34: see ?regex
sub("\\s+$", "", str, perl = TRUE) ## PCRE-style white space

## capitalizing
txt <- "a test of capitalizing"
gsub("(\\w)(\\w*)", "\\U\\1\\L\\2", txt, perl=TRUE)
gsub("\\b(\\w)", "\\U\\1", txt, perl=TRUE)

txt2 <- "useRs may fly into JFK or laGuardia"
gsub("(\\w)(\\w*)(\\w)", "\\U\\1\\E\\2\\U\\3", txt2, perl=TRUE)
sub("(\\w)(\\w*)(\\w)", "\\U\\1\\E\\2\\U\\3", txt2, perl=TRUE)

## named capture
notables <- c(" Ben Franklin and Jefferson Davis",
              "\tMillard Fillmore")
# name groups 'first' and 'last'
name.rex <- "(?<first>[[:upper:]]+[[:lower:]]+)(?<last>[[:upper:]]+[[:lower:]]+)"
(parsed <- regexpr(name.rex, notables, perl = TRUE))
gregexpr(name.rex, notables, perl = TRUE)[[2]]
parse.one <- function(res, result) {
  m <- do.call(rbind, lapply(seq_along(res), function(i) {
    if(result[i] == -1) return("")
    st <- attr(result, "capture.start")[i, ]
    substr(res[i], st, st + attr(result, "capture.length")[i, ] - 1)
  }))
  colnames(m) <- attr(result, "capture.names")
  m
}
parse.one(notables, parsed)

## Decompose a URL into its components.
## Example by LT (http://www.cs.uiowa.edu/~luke/R/regex.html).
x <- "http://stat.umn.edu:80/xyz"
m <- regexec("^(([[:^:]]+://)?([[:^:]]+)(:([0-9]+))?(/.*))", x)
m
regmatches(x, m)
## Element 3 is the protocol, 4 is the host, 6 is the port, and 7
## is the path. We can use this to make a function for extracting the
## parts of a URL:
URL_parts <- function(x) {
  m <- regexec("^(([[:^:]]+://)?([[:^:]]+)(:([0-9]+))?(/.*))", x)
  parts <- do.call(rbind,
                  lapply(regmatches(x, m), `[,`, c(3L, 4L, 6L, 7L)))
  colnames(parts) <- c("protocol", "host", "port", "path")
  parts
}
URL_parts(x)

## gregexec() may match multiple times within a single string.
pattern <- "([[:alpha:]]+)([[:digit:]]+)"
s <- "Test: A1 BC23 DEF456"
m <- gregexec(pattern, s)
m
regmatches(s, m)

## Before gregexec() was implemented, one could emulate it by running
## regexec() on the regmatches obtained via gregexpr(). E.g.:

```



```
lapply(regmatches(s, gregexpr(pattern, s)),
       function(e) regmatches(e, regexec(pattern, e)))
```

grepRaw

grepRawpatternx

```
grepRaw(pattern, x, offset = 1L, ignore.case = FALSE,
        value = FALSE, fixed = FALSE, all = FALSE, invert = FALSE)
```

pattern	fixed = TRUEcharToRaw
x	charToRaw
ignore.case	FALSETRUE
offset	"^"
value	
fixed	TRUEpattern
all	TRUE
invert	TRUEvalue = TRUE

```
grepxall = TRUE
invertvalue = TRUEoffsetinvert = TRUEall = TRUEex"^"$"
fixed = TRUEvalue = TRUE
```

```
grepRaw(value = FALSE)all = FALSE
grepRaw(value = TRUE, all = FALSE)x
grepRaw(value = TRUE, all = TRUE)
```

ignore.casevaluefixedallinvert

<https://github.com/laurikari/tre/fixed> = TRUE

regexp
grep

```
grepRaw("no match", "textText") # integer(0): no match
grepRaw("adf", "adadfadfdfadadf") # 3 - the first match
grepRaw("adf", "adadfadfdfadadf", all=TRUE, fixed=TRUE)
## [1] 3 6 13 -- three matches
```

groupGeneric

MathOpsmatrixOpsSummaryComplex

S3 methods for group generics have prototypes:

Math(x, ...)
Ops(e1, e2)
Complex(z)
Summary(..., na.rm = FALSE)
matrixOps(x, y)

xyze1e2

...

na.rm

"Math""Ops""Summary""matrixOps""Complex"[factordata](#)[.framed](#)[ifft](#)[timeordered](#)Ops
[POSIXt](#)DateMathOps[package_version](#)OpsSummarytsOps

"Math"

abssignsqrt
floorceilingtrunc
roundsignif
explogexpm1log1p
cossintan
cospisinpitani
acosasinatan
coshsinhtanh
acoshasinhatanh
lgammagammadigammatrigamma
cumsumcumprodcummaxcummin

xlogroundsigniftrunc

"Ops"

"+" "-" "x" "/" "^" "%" "%/"
"&" "|" "!"
"==" "!=" "<" "<=" ">" ">="

+ - ! Opse2

OpschooseOpsMethod()?chooseOpsMethodchooseOpsMethod()

[data.frame](#)"Compare"=="<"Logic"&|![matrix](#)

"matrixOps"

```

    "%*%"
    %*%crossprod()tcrossprod()Ops
    "Summary"
        allany
        sumprod
        minmax
        range

    data.frame"Summary""Math"x
        is.numeric(x) || is.logical(x) || is.complex(x)
    "Complex"
        ArgConjImModRe
    z

"class"oldClassclassOps.integer
"Math"

```

```

.GenericUseMethod

```

```

Ops.Method""
.Group""

```

methods

```

require(utils)

d.fr <- data.frame(x = 1:9, y = stats::rnorm(9))
class(1 + d.fr) == "data.frame" ##-- add to d.f. ...

methods("Math")
methods("Ops")
methods("Summary")
methods("Complex") # none in base R

```

grouping

grouping

grouping(...)

...

NA

"radix"[order](#)2³¹

[order](#)[xtfrm](#)

"grouping"

ends

maxgrpn

[order](#)[xtfrm](#)

```
(ii <- grouping(x <- c(1, 1, 3:1, 1:4, 3), y <- c(9, 9:1), z <- c(2, 1:9)))  
## 6 5 2 1 7 4 10 8 3 9  
rbind(x, y, z)[, ii]
```

gzcon

gzcongzzip

gzcon(con, level = 6, allowNonCompressed = TRUE, text = FALSE)

con

level

allowNonCompressed

text TRUE[pushBackreadBin](#)

```
con
gzipallowNonCompressed
contextConnectionopen = "w"rawConnection
```

```
"connection"
```

```
gzfile
```

```
## Uncompress a data file from a URL
z <- gzcon(url("https://www.stats.ox.ac.uk/pub/datasets/csb/ch12.dat.gz"))
# read.table can only read from a text-mode connection.
raw <- textConnection(readLines(z))
close(z)
dat <- read.table(raw)
close(raw)
dat[1:4, ]
```

```
## gzfile and gzcon can inter-work.
## Of course here one would use gzfile, but file() can be replaced by
## any other connection generator.
zzfil <- tempfile(fileext = ".gz")
zz <- gzfile(zzfil, "w")
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = zz, sep = "\n")
close(zz)
readLines(zz <- gzcon(file(zzfil, "rb")))
close(zz)
unlink(zzfil)
```

```
zzfil2 <- tempfile(fileext = ".gz")
zz <- gzcon(file(zzfil2, "wb"))
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = zz, sep = "\n")
close(zz)
readLines(zz <- gzfile(zzfil2))
close(zz)
unlink(zzfil2)
```

```
hexmode
```

```

as.hexmode(x)

## S3 method for class 'hexmode'
as.character(x, keepStr = FALSE, ...)

## S3 method for class 'hexmode'
format(x, width = NULL, upper.case = FALSE, ...)

## S3 method for class 'hexmode'
print(x, ...)

x          "hexmode"
keepStr    logical TRUE
width      NULL
upper.case
...

"hexmode"[
as.character(x)attributeskeepStr=TRUEdimdimnamesnamesformat()width = NULL
as.hexmode"integer""double"0-9a-fA-FNA"hexmode"
!|&

octmodesprintfstrtoi

i <- as.hexmode("7fffffff")
i; class(i)
identical(as.integer(i), .Machine$integer.max)

hm <- as.hexmode(c(NA, 1)); hm
as.integer(hm)

Xm <- as.hexmode(1:16)
Xm # print()s via format()
stopifnot(nchar(format(Xm)) == 2)
Xm[-16] # *no* leading zeroes!
stopifnot(format(Xm[-16]) == c(1:9, letters[1:6]))

## Integer arithmetic (remaining "hexmode"):
16*Xm
Xm^2
-Xm
(fac <- factorial(Xm[1:12])) # !1, !2, !3, !4 .. in hexadecimal
as.integer(fac) # indeed the same as factorial(1:12)

```

Hyperbolic

cosh(x)
sinh(x)
tanh(x)
acosh(x)
asinh(x)
atanh(x)

x

Math

asin

Math

cossintanacosasinatan
plogistanh()x

iconv

iconv(x, from = "", to = "", sub = NA, mark = TRUE, toRaw = FALSE)

iconvlist()

x [as.character](#)NULLrawiconv(toRaw = TRUE)

from

to

sub NA"byte""<xx>""Unicode""<U+xxxx>"c99"\uxxxx""\Uxxxxxxxx"

mark

toRaw

```
""latin1""UTF-8"
iconvlistmusliconv(5)iconv
xNANULLtoRaw = TRUEsub
iconv//TRANSLITto
"ASCII""C""POSIX""ASCII//TRANSLIT""ASCII"sub = "c99"sub = "?"ASCII"*
xEncodingfrom = ""from
iconvfrom
sub = "Unicode"sub = "c99"sub = "byte"
```

```
toRaw = FALSExNA_character_NA_character_NA_character_
mark = TRUEto"latin1""UTF-8"to = ""
toRaw = TRUExNULLNA_character_
iconvlist()
```

```
iconvgliblibiconvwin_iconviconvlist//TRANSLITwin_iconvto = "ASCII"
libiconvlibiconv?NA
iconvlibiconv
muslglbclib//TRANSLIT*
//TRANSLIT
to = "ASCII"
libiconvto = "C99"\uxxxx\Uxxxxxxxxxx
```

```
U+FEFFconnectionx
from"UCS-2""UTF-16""UTF-32"xfromU+FEFF
```

```
"iso885915""latin-9""latin9"libiconv
"utf8""mac""macroman""utf8""UTF-8"fromtoiconvfileEncoding"macintosh"https://en.wikipedia.org/wiki/Mac\_OS\_Roman
subsub = "c99"sub = "Unicode"
```

```
localeToCharsetfile
```



```

## In principle, as not all systems have iconvlist
try(utils::head(iconvlist(), n = 50))

## Not run:
## convert from Latin-2 to UTF-8: two of the glibc iconv variants.
iconv(x, "ISO_8859-2", "UTF-8")
iconv(x, "LATIN2", "UTF-8")

## End(Not run)

## Both x below are in latin1 and will only display correctly in a
## locale that can represent and display latin1.
x <- "fran\xE7ais"
Encoding(x) <- "latin1"
x
charToRaw(xx <- iconv(x, "latin1", "UTF-8"))
xx

## The results in the comments are those from glibc and GNU libiconv
iconv(x, "latin1", "ASCII")      # NA
iconv(x, "latin1", "ASCII", "?") # "fran?ais"
iconv(x, "latin1", "ASCII", "")  # "franais"
iconv(x, "latin1", "ASCII", "byte") # "fran<e7>ais"
iconv(xx, "UTF-8", "ASCII", "Unicode") # "fran<U+00E7>ais"
iconv(xx, "UTF-8", "ASCII", "c99")  # "fran\\u00e7ais"

## Extracts from old R help files (they are nowadays in UTF-8)
x <- c("Ekstr\xfbm", "J\xfbreskog", "bi\xdfchen Z\xfccher")
Encoding(x) <- "latin1"
x
try(iconv(x, "latin1", "ASCII//TRANSLIT")) # platform-dependent
## glibc gives "Ekstroem" "Joreskog" "bisschen Zurcher"
## macOS 14 gives "Ekstrom" "J\"oreskog" "bisschen Z\"urcher"
## musl gives "Ekstr*m" "J*reskog" "bi*chen Z*rcher"
iconv(x, "latin1", "ASCII", sub = "byte")

## and for Windows' 'Unicode'
str(xx <- iconv(x, "latin1", "UTF-16LE", toRaw = TRUE))
iconv(xx, "UTF-16LE", "UTF-8")

emoji <- "\U0001f604"
iconv(emoji, "latin1", sub = "Unicode") # "<U+1F604>"
iconv(emoji, "latin1", sub = "c99")

```

icuSetCollate

```
icuSetCollate(...)
```

```
icuGetCollate(type = c("actual", "valid"))
```

```
...
type          "actual""valid"
```

<https://icu.unicode.org/icuSetCollate>

```
locale "da_DK"
case_first "upper""lower""default"
alternate_handling "non_ignorable""shifted"
strength "primary""secondary""tertiary""quaternary""identical"
french_collation "on""off""default"
normalization "on""off"
case_level "on""off"
hiragana_quaternary "on""off"
```

```
locale
"none"
"ASCII" strcmp
"default" R_ICU_LOCALELC_ALLLC_COLLATE
""root" https://www.unicode.org/reports/tr35/tr35-collation.html#Root\_Collation
https://unicode-org.github.io/icu/userguide/locale/icuGetCollate\("actual"\)
"root""root""en_GB""C"R_ICU_LOCALE"C"
case_level = "on", strength = "primary"alternate_handling = "shifted"
CR_ICU_LOCALElocalelocale = "none"Sys.setlocale"C"LC_ALLLC_COLLATER_ICU_LOCALE
R_ICU_LOCALESys.setlocaleLC_ALLLC_COLLATER_ICU_LOCALER_ICU_LOCALE
localeSys.setlocale
```

```
icuGetCollate"ICU not in use""da""da_DK""root"
```

```
R_ICU_LOCALEicuSetCollateicuSetCollate(locale = "default")LC_COLLATE
```

```
sort
capabilitiesextSoftVersion
https://unicode-org.github.io/icu/userguide/collation/
```

```
## These examples depend on having ICU available, and on the locale.
## As we don't know the current settings, we can only reset to the default.
if(capabilities("ICU")) withAutoprint({
  icuGetCollate()
  icuGetCollate("valid")
  x <- c("Aarhus", "aarhus", "safe", "test", "Zoo")
  sort(x)
  icuSetCollate(case_first = "upper"); sort(x)
  icuSetCollate(case_first = "lower"); sort(x)

  ## Danish collates upper-case-first and with 'aa' as a single letter
  icuSetCollate(locale = "da_DK", case_first = "default"); sort(x)
  ## Estonian collates Z between S and T
  icuSetCollate(locale = "et_EE"); sort(x)
  icuSetCollate(locale = "default"); icuGetCollate("valid")
})
```

identical

TRUEFALSE

```
identical(x, y, num.eq = TRUE, single.NA = TRUE, attrib.as.set = TRUE,
  ignore.bytecode = TRUE, ignore.environment = FALSE,
  ignore.srcref = TRUE, extptr.as.ref = FALSE)
```

```
xy
num.eq          doublecomplexNA==0+0i
single.NA       NANaNsingl.NA = FALSE
attrib.as.set   attributesxslotattrib.as.set = FALSE
ignore.bytecode

ignore.environment

ignore.srcref   "srcref"
extptr.as.ref
```

```
identicalifwhile&&|
==!=xyFALSENANAif(x == y)....
all.equal
identicalidentical
single.NAidenticalNaNNA_real_NaNNA
"bytes""bytes"
```

```

attrib.as.set
ignore.bytecodecmpfundisassemble()
identical"POSIXlt"

```

```

    identical(x, y,
              num.eq = FALSE, single.NA = FALSE, attrib.as.set = FALSE,
              ignore.bytecode = FALSE, ignore.environment = FALSE,
              ignore.srcref = FALSE, extptr.as.ref = TRUE)

```

```

TRUEFALSENA

```

all.equalComparisonLogic

```

identical(1, NULL) ## FALSE -- don't try this with ==
identical(1, 1.) ## TRUE in R (both are stored as doubles)
identical(1, as.integer(1)) ## FALSE, stored as different types

```

```

x <- 1.0; y <- 0.999999999999
## how to test for object equality allowing for numeric fuzz :
(E <- all.equal(x, y))
identical(TRUE, E)
isTRUE(E) # alternative test
## If all.equal thinks the objects are different, it returns a
## character string, and the above expression evaluates to FALSE

```

```

## even for unusual R objects :
identical(.GlobalEnv, environment())

```

```

### ----- Pickyness Flags : -----

```

```

## the infamous example:
identical(0., -0.) # TRUE, i.e. not differentiated
identical(0., -0., num.eq = FALSE)
## similar:
identical(NaN, -NaN) # TRUE
identical(NaN, -NaN, single.NA = FALSE) # differ on bit-level

```

```

### For functions ("closure"s): -----
### ~~~~~~
f <- function(x) x
f
g <- compiler::cmpfun(f)
g

```

```

identical(f, g)                                # TRUE, as bytecode is ignored by default
identical(f, g, ignore.bytecode=FALSE) # FALSE: bytecode differs

## GLM families contain several functions, some of which share an environment:
p1 <- poisson() ; p2 <- poisson()
identical(p1, p2)                                # FALSE
identical(p1, p2, ignore.environment=TRUE) # TRUE

## in interactive use, the 'keep.source' option is typically true:
op <- options(keep.source = TRUE) # and so, these have differing "srcfile" :
f1 <- function() {}
f2 <- function() {}
identical(f1,f2)# ignore.srcfile= TRUE : TRUE
identical(f1,f2, ignore.srcfile=FALSE)# FALSE
options(op) # revert to previous state

```

identity

identity(x)

x

diag

ifelse

ifelsetestyesnotestTRUEFALSE

ifelse(test, yes, no)

test

yes test

no test

yesnoyestestno

test

```
"class"testyesnoyesno
```

```
testoldClass testyesno
```

```
(tmp <- yes; tmp[!test] <- no[!test]; tmp)
```

```
test
```

```
if(test) yes else noifelse(test, yes, no)testlength(test) == 1  
srcrfttestyessrcrftifelseyestestnoif(test) yes else noyesno
```

```
if
```

```
x <- c(6:-4)  
sqrt(x) #- gives warning  
sqrt(ifelse(x >= 0, x, NA)) # no warning
```

```
## Note: the following also gives the warning !  
ifelse(x >= 0, sqrt(x), NA)
```

```
## ifelse() strips attributes  
## This is important when working with Dates and factors  
x <- seq(as.Date("2000-02-29"), as.Date("2004-10-04"), by = "1 month")  
## has many "yyyy-mm-29", but a few "yyyy-03-01" in the non-leap years  
y <- ifelse(as.POSIXlt(x)$mday == 29, x, NA)  
head(y) # not what you expected ... ==> need restore the class attribute:  
class(y) <- class(x)  
y  
## This is a (not atypical) case where it is better *not* to use ifelse(),  
## but rather the more efficient and still clear:  
y2 <- x  
y2[as.POSIXlt(x)$mday != 29] <- NA  
## which gives the same as ifelse()+class() hack:  
stopifnot(identical(y2, y))
```

```
## example of different return modes (and 'test' alone determining length):  
yes <- 1:3  
no <- pi^(1:4)  
utils::str( ifelse(NA, yes, no) ) # logical, length 1  
utils::str( ifelse(TRUE, yes, no) ) # integer, length 1  
utils::str( ifelse(FALSE, yes, no) ) # double, length 1
```

integer

"integer"

```
integer(length = 0)
as.integer(x, ...)
is.integer(x)
```

length

x

...

$\pm 2 \times 10^9$ double

integer0

```
as.integerNANAas.integer(x) trunc(x)0x0Xas.vectorxstorage.mode(x) <- "integer"
is.integerTRUEFALSEFALSE
```

```
is.integer(x)xroundis.wholenumber(x)
```

numericstorage.mode

roundceilingfloor

```
## as.integer() truncates:
x <- pi * c(-1:1, 10)
as.integer(x)
```

```
is.integer(1) # is FALSE !
```

```
is.wholenumber <-
  function(x, tol = .Machine$double.eps^0.5) abs(x - round(x)) < tol
is.wholenumber(1) # is TRUE
(x <- seq(1, 5, by = 0.5) )
is.wholenumber( x ) #--> TRUE FALSE TRUE ...
```

interaction

interactioninteraction

interaction(..., drop = FALSE, sep = ".", lex.order = FALSE)

...

drop dropTRUE

sep

lex.order

sep.

lex.order = FALSElex.order = TRUE:

factor:f:ginteraction(f, g, sep = ":")fg

a <- gl(2, 4, 8)

b <- gl(2, 2, 8, labels = c("ctrl", "treat"))

s <- gl(2, 1, 8, labels = c("M", "F"))

interaction(a, b)

interaction(a, b, s, sep = ":")

stopifnot(identical(a:s,

 interaction(a, s, sep = ":", lex.order = TRUE)),

 identical(a:s:b,

 interaction(a, s, b, sep = ":", lex.order = TRUE)))

interactive

TRUEFALSE

interactive()


```
Rterm.exestdinstdin--interactive--essRterm.exereadline--interactive
```

```
stopoptions("showWarnCalls")
```

```
--save--no-save--vanilla
```

```
dev.newoptions("device")
```

```
interactive()helpdebuggerinstall.packages
```

```
source.First
```

```
.First <- function() if(interactive()) x11()
```

Internal

```
.Internal
```

```
.Internal(call)
```

```
call
```

```
.Primitive.External
```

InternalMethods

methods

```
[[[extract_itex]<-[[[/extract_itex]<-  
lengthlength<-lengthsdimnamesdimnames<-dimdim<-namesnames<-levels<-@@<-  
cunlistcbindrbind  
as.characteras.complexas.doubleas.integeras.logicalas.rawas.vectoras.call  
as.environmentis.arrayis.matrixis.naanyNAis.nanis.finiteis.infiniteis.numeric  
ncharrep.intrep_lenseq.int"seq"is.unsortedxtfrm  
is.nameis.symbolas.numericas.doubleas.doubleas.numeric
```

```
.S3PrimitiveGenerics.internalGenerics.Internal(..)
```

is.object

methods

invisible

```
invisible(x = NULL)
```

```
x          NULL
```

withVisiblreturnfunction

```
# These functions both return their argument  
f1 <- function(x) x  
f2 <- function(x) invisible(x)  
f1(1) # prints  
f2(1) # does not
```

`is.finite`

`is.finite``is.infinite`
`Inf``-Inf``NaN``Inf``NaN``NA`

`is.finite(x)`
`is.infinite(x)`
`is.nan(x)`

`Inf`
`NaN`

`x`

`is.finite``TRUE``FALSE``TRUE``FALSE``TRUE``FALSE`
`is.infinite``TRUE``FALSE``TRUE``FALSE`
`is.nan``TRUE``FALSE``TRUE``FALSE`
`is.nan``TRUE``FALSE``TRUE``FALSE`
`is.nan``TRUE``FALSE``TRUE``FALSE`

`xdim``dimnames``names`

`Arithmetic``+``-` `Inf``NaN`
`Inf`
`NaN``NaN``NA`

<https://en.wikipedia.org/wiki/NaN>

https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html
`isfinite``is.finite`

`NA`
`Arithmetic``double`

```
pi / 0 ## = Inf a non-zero number divided by zero creates infinity
0 / 0 ## = NaN

1/0 + 1/0 # Inf
1/0 - 1/0 # NaN

stopifnot(
  1/0 == Inf,
  1/Inf == 0
)
sin(Inf)
cos(Inf)
tan(Inf)
```

is.function

```
is.function(x)
is.primitive(x)
```

```
x
```

```
is.primitive(x)xtypeof(x)"builtin""special"
```

```
TRUExFALSE
```

```
is.function(1) # FALSE
is.function (is.primitive) # TRUE: it is a function, but ..
is.primitive(is.primitive) # FALSE: it's not a primitive one, whereas
is.primitive(is.function) # TRUE: that one *is*
```

is.language

```
is.languageTRUExnamecallexpression
```

```
is.language(x)
```

```
x
```

`name``typeof``is.symbol`

`typeof(x) == "language"``is.language(x)``is.language(y)`

```
ll <- list(a = expression(x^2 - 2*x + 1), b = as.name("Jim"),
          c = as.expression(exp(1)), d = call("sin", pi))
sapply(ll, typeof)
sapply(ll, mode)
stopifnot(sapply(ll, is.language))
```

`is.object`

`TRUE``OBJECT``FALSE``OBJECT``"class"`

`is.atomic``attributes(x)"class"`

`is.object(x)`

`x`

`classmethods`

`isS4`

`is.object(1) # FALSE`
`is.object(as.factor(1:3)) # TRUE`

`is.recursive`

```
is.atomicTRUExFALSEis.atomic(NULL)FALSE
is.recursiveTRUExFALSE
```

```
is.atomic(x)
is.recursive(x)
```

x

```
is.atomic"logical""integer""numeric""complex""character""raw"
NULLas.nameS4typeof
is.vector
```

```
is.listis.languagedemo("is.things")
```

```
require(stats)
```

```
is.a.r <- function(x) c(is.atomic(x), is.recursive(x))

is.a.r(c(a = 1, b = 3)) # TRUE FALSE
is.a.r(list())          # FALSE TRUE - a list is a list
is.a.r(list(2))         # FALSE TRUE
is.a.r(lm)              # FALSE TRUE
is.a.r(y ~ x)           # FALSE TRUE
is.a.r(expression(x+1)) # FALSE TRUE
is.a.r(quote(exp))      # FALSE FALSE
is.a.r(NULL)            # FALSE FALSE

# Reproduce pre-4.4 behavior of is.atomic()
is.atomicN <- function(x) is.atomic(x) || is.null(x)
is.atomicN(NULL) # TRUE
```

`is.single`

`is.single``is.single(x)``x`

`is.unsorted`

`is.unsorted(x, na.rm = FALSE, strictly = FALSE)``x``na.rm``strictly``is.unsorted``NA=>x[i]x[i-1]i2:length(x)is.unsorted()``sortorder`

ISOdatetime

```
ISOdatetime(year, month, day, hour, min, sec, tz = "")  
ISOdate(year, month, day, hour = 12, min = 0, sec = 0, tz = "GMT")
```

```
yearmonthday  
hourminsec  
tz          "" "GMT"
```

```
ISOdatetimeISOdatestrptimeISOdate"Date"
```

```
strptime0:9999
```

```
"POSIXct"
```

```
strptime
```

isS4

```
isS4(object)
```

```
asS4(object, flag = TRUE, complete = TRUE)  
asS3(object, flag = TRUE, complete = TRUE)
```

```
object  
flag  
complete
```

```
isS4require  
asS3completeobjectcompleteTRUEs3PartarraymatrixNULL  
isS4
```



```
isS4TRUEFALSE
asS4asS3asS3.S3ClassasS3"S4"completeFALSE
```

[is.object](#)

```
isS4(pi) # FALSE
isS4(getClass("MethodDefinition")) # TRUE
```

[isSymmetric](#)

```
objectcomplexZisSymmetric(Z)isSymmetric(Z, trans = "T")
```

```
isSymmetric(object, ...)
## S3 method for class 'matrix'
isSymmetric(object, tol = 100 * .Machine$double.eps,
             tol1 = 8 * tol, trans = "C", ...)
```

object	matrix
tol	all.equal.numeric
tol1	<code>isSymmetric.matrix()</code>
trans	charactercomplexZ"C"Conj(t(Z))Z"T"t(Z)Z
...	all.equal <code>objectcheck.attributes = FALSE</code>

[matrixeigenall.equal](#)

```
mrownamescolnamesunnname(m)
```

object

[eigen](#)`isSymmetric``symmetric`

```

isSymmetric(D3 <- diag(3)) # -> TRUE

D3[2, 1] <- 1e-100
D3
isSymmetric(D3) # TRUE
isSymmetric(D3, tol = 0) # FALSE for zero-tolerance

## Complex Matrices - Hermitian or not
z <- sqrt(matrix(-1:2 + 0i, 2)); Z <- t(Conj(z)) %*% z
ZtZ <- t(z) %*% z
Z ; ZtZ
isSymmetric(Z)      # TRUE
isSymmetric(Z + 1)  # TRUE
isSymmetric(Z + 1i) # FALSE -- a Hermitian matrix has a *real* diagonal

colnames(D3) <- c("X", "Y", "Z")
isSymmetric(D3)      # FALSE (as row and column names differ)
isSymmetric(D3, check.attributes=FALSE) # TRUE (as names are not checked)

```

jitter

```

jitter(x, factor = 1, amount = NULL)

x
factor
amount      = 0factor * z/50
             NULLfactor * d/5dx

rr <- x + runif(n, -a, a)n <- length(x)aamount
z <- max(x) - min(x)aamountz
amount == 0a <- factor * z/50
amountNULLa <- factor * d/5x

jitter(x, ...)xamount

```

rugjitter

```
round(jitter(c(rep(1, 3), rep(1.2, 4), rep(3, 3))), 3)
## These two 'fail' with S-plus 3.x:
jitter(rep(0, 7))
jitter(rep(10000, 5))
```

kappa

kappa()*RQR*

rcond()

```
kappa(z, ...)
## Default S3 method:
kappa(z, exact = FALSE,
      norm = NULL, method = c("qr", "direct"),
      inv_z = solve(z),
      triangular = FALSE, uplo = "U", ...)
```

```
## S3 method for class 'lm'
```

```
kappa(z, ...)
```

```
## S3 method for class 'qr'
```

```
kappa(z, ...)
```

```
.kappa_tri(z, exact = FALSE, LINPACK = TRUE, norm = NULL, uplo = "U", ...)
```

```
rcond(x, norm = c("O", "I", "1"), triangular = FALSE, uplo = "U", ...)
```

zx qr"lm"

exact

norm norm()kappa()"2"rcond()"0".kappa_tri()exact"2""0"exact=FALSE"I"
exact=TRUE"2"typenorm(., type = *)

method "qr"

inv_z exact=TRUE, norm != "2"solve(z)zsolve(z)exact

triangular zx

uplo "U""L"triangular = TRUE

LINPACK zdtrco()

... kappa.*()LINPACKnorm"2"

```

kappa()exact = FALSE $QR(x)z = QRsvd$ 
method = "direct"rcond()kappa*()exact
kapparcond
.kappa_trikappa.qrkappa.defaultttriuplo = "U""L""U"

```

*kappa*exact = FALSE

DTRCONZTRCONDTRCO

<https://netlib.org/lapack/><https://netlib.org/linpack/>

https://netlib.org/lapack/lug/lapack_lug.html

$normsvdqrQR$

```

kappa(x1 <- cbind(1, 1:10)) # 15.71
kappa(x1, exact = TRUE)    # 13.68
kappa(x2 <- cbind(x1, 2:11)) # high! [x2 is singular!]

hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
sv9 <- svd(h9 <- hilbert(9))$ d
kappa(h9) # pretty high; by default {exact=FALSE, method="qr"} :
kappa(h9) == kappa(qr.R(qr(h9)), norm = "1")
all.equal(kappa(h9, exact = TRUE), # its definition:
           max(sv9) / min(sv9),
           tolerance = 1e-12) ## the same (typically down to 2.22e-16)
kappa(h9, exact = TRUE) / kappa(h9) # 0.677 (i.e., rel.error = 32%)

## Exact kappa for rectangular matrix
## panmagic.6npm1(7) :
pm7 <- rbind(c( 1, 13, 18, 23, 35, 40, 45),
             c(37, 49,  5, 10, 15, 27, 32),
             c(24, 29, 41, 46,  2, 14, 19),
             c(11, 16, 28, 33, 38, 43,  6),
             c(47,  3,  8, 20, 25, 30, 42),
             c(34, 39, 44,  7, 12, 17, 22),
             c(21, 26, 31, 36, 48,  4,  9))

```

```

kappa(pm7, exact=TRUE, norm="1") # no problem for square matrix

m76 <- pm7[,1:6]
(m79 <- cbind(pm7, 50:56, 63:57))

## Moore-Penrose inverse { ~= MASS::ginv(); differing tol (value & meaning)}:
## pinv := p(seudo) inv(erse)
pinv <- function(X, s = svd(X), tol = 64*.Machine$double.eps) {
  if (is.complex(X))
    s$u <- Conj(s$u)
  dx <- dim(X)
  ## X = U D V' ==> Result = V {1/D} U'
  pI <- function(u,d,v) tcrossprod(v, u / rep(d, each = dx[1L]))
  pos <- (d <- s$d) > max(tol * max(dx) * d[1L], 0)
  if (all(pos))
    pI(s$u, d, s$v)
  else if (!any(pos))
    array(0, dx[2L:1L])
  else { # some pos, some not:
    i <- which(pos)
    pI(s$u[, i, drop = FALSE], d[i],
      s$v[, i, drop = FALSE])
  }
}

## rectangular
kappa(m76, norm="1")
try( kappa(m76, exact=TRUE, norm="1") )# error in solve().. must be square

## ==> use pseudo-inverse instead of solve() for rectangular {and norm != "2"}:
iZ <- pinv(m76)
kappa(m76, exact=TRUE, norm="1", inv_z = iZ)
kappa(m76, exact=TRUE, norm="M", inv_z = iZ)
kappa(m76, exact=TRUE, norm="I", inv_z = iZ)

iX <- pinv(m79)
kappa(m79, exact=TRUE, norm="1", inv_z = iX)
kappa(m79, exact=TRUE, norm="M", inv_z = iX)
kappa(m79, exact=TRUE, norm="I", inv_z = iX)

## Using a more "accurate" than default inv_z [example by Cleve Moler]:
A <- rbind(c(4.1, 2.8),
           c(9.676, 6.608))
kappa(A) # -> Inf
kappa(A, exact=TRUE) # 8.675057e+15 ( 2-norm )

## now for the 1-norm :
try(kappa(A, exact=TRUE, norm = "1")) #-> Error: computationally singular
try(kappa(A, exact=TRUE, norm = "1",
          inv_z = solve(A, tol = 1e-19))) # 5.22057e16 on x86_64 Linux with GCC

```

kronecker

XY

```
kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)
X %x% Y
```

X

Y

FUN

make.dimnames XY

... FUN

XYXFUN(x, Y, ...)

%x%kroneckerFUN"*"

Adim(X) * dim(Y)

[outer](#)kronecker%*%

```
# simple scalar multiplication
( M <- matrix(1:6, ncol = 2) )
kronecker(4, M)
# Block diagonal matrix:
kronecker(diag(1, 3), M)

# ask for dimnames

fred <- matrix(1:12, 3, 4, dimnames = list(LETTERS[1:3], LETTERS[4:7]))
bill <- c("happy" = 100, "sad" = 1000)
kronecker(fred, bill, make.dimnames = TRUE)

bill <- outer(bill, c("cat" = 3, "dog" = 4))
kronecker(fred, bill, make.dimnames = TRUE)
```

l10n_info

l10n_info()

codepagesystem.codepage

MBCS

UTF-8

Latin-1

codeset ""

codepage
system.codepage

[Sys.getlocalelocaleconv](#)

l10n_info()

labels

labels(object, ...)

object

...

seq_along(x)NULLseq_len(d[i])

lapply

```
lapplyXFUNX
sapplylapplysimplify = "array"simplify2array()sapply(x, f, simplify = FALSE,
USE.NAMES = FALSE)lapply(x, f)
vapplysapply
replicatesapply
simplify2array()sapply()simplifymapply()
```

```
lapply(X, FUN, ...)
sapply(X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
vapply(X, FUN, FUN.VALUE, ..., USE.NAMES = TRUE)
replicate(n, expr, simplify = "array")
simplify2array(x, higher = TRUE, except = c(0L, 1L))
```

X	<code>expressionbase::as.list</code>
FUN	<code>X+%*%</code>
...	<code>FUN</code>
simplify	<code>sapplyTRUEsimplify = "array"array=length(dim(.))FUN(X[[i]])</code>
USE.NAMES	<code>TRUEEXXnames...</code>
FUN.VALUE	
n	
expr	
x	<code>lapply()</code>
higher	<code>simplify2array()higher = FALSEsapply(*, simplify = "array") simplify = TRUE</code>
except	<code>NULLc(0L, 1L)sapply0:1NULL</code>

```
FUNmatch.funlapply
FUNXFUNX
...FUNXFUN...FUNXFUN...
sapplyXXX
vapplyFUNFUN.VALUE
lapplyvapplyas.list
```



```

lapply(sapply(simplify = FALSE)replicate(simplify = FALSE)
sapply(simplify = TRUE)replicate(simplify = TRUE)Xn = 0Xnreplicate
vapplyFUN.VALUElength(FUN.VALUE) == 1XFUN.VALUEarraylength(FUN.VALUE)length(X)a
dim(a) == c(dim(FUN.VALUE), length(X))
FUN.VALUEXsapply

```

```

sapply(*, simplify = FALSE, USE.NAMES = FALSE)lapply(*)
lapply(bquote(FUN(X[[i]], ...))ifUNsys.callmatch.calllapply(11, function(x)
is.numeric(x))is.numeric
expr...replicate

```

[apply](#)[tapply](#)[mapply](#)[rapply](#)[lapply\(\)](#)[eapply](#)[environment](#)

```

require(stats); require(graphics)

x <- list(a = 1:10, beta = exp(-3:3), logic = c(TRUE,FALSE,FALSE,TRUE))
# compute the list mean for each list element
lapply(x, mean)
# median and quartiles for each list element
lapply(x, quantile, probs = 1:3/4)
sapply(x, quantile)
i39 <- sapply(3:9, seq) # list of vectors
sapply(i39, fivenum)
vapply(i39, fivenum,
       c(Min. = 0, "1st Qu." = 0, Median = 0, "3rd Qu." = 0, Max. = 0))

## sapply(*, "array") -- artificial example
(v <- structure(10*(5:8), names = LETTERS[1:4]))
f2 <- function(x, y) outer(rep(x, length.out = 3), y)
(a2 <- sapply(v, f2, y = 2*(1:5), simplify = "array"))
a.2 <- vapply(v, f2, outer(1:3, 1:5), y = 2*(1:5))
stopifnot(dim(a2) == c(3,5,4), all.equal(a2, a.2),
          identical(dimnames(a2), list(NULL,NULL,LETTERS[1:4])))

hist(replicate(100, mean(rexp(10))))

## use of replicate() with parameters:
foo <- function(x = 1, y = 2) c(x, y)
# does not work: bar <- function(n, ...) replicate(n, foo(...))
bar <- function(n, x) replicate(n, foo(x = x))
bar(5, x = 3)

```

Last.value

.Last.valuepackage:base

.Last.value

.Last.value

.Last.valuepackage:base

[eval](#)

```
## These will not work correctly from example(),  
## but they will in make check or if pasted in,  
## as example() does not run them at the top level  
gamma(1:15)      # think of some intensive calculation...  
fac14 <- .Last.value # keep them
```

```
library("splines") # returns invisibly  
.Last.value      # shows what library(.) above returned
```

La_library

LAPACK

La_library()

""LAPACKLAPACKlibRlapack.solibRlapack.dyliblibRlapack.solibRlapack.dylibLAPACK

ILAVR

[extSoftVersion](#)BLAS

[La_version](#)

La_library()

La_version

La_version()

DLARTG Dlassq ZLARTG Zlassq

extSoftVersion

La_library

La_version()

length

length(x)
length(x) <- value

x
value

length<-"factor"
NA NULL

lengthinteger $2^{31} - 1$
NULL
x""

double

ncharlengths

```
length(diag(4)) # = 16 (4 x 4)
length(options()) # 12 or more
length(y ~ x1 + x2 + x3) # 3
length(expression(x, {y <- x^2; y+2}, x^y)) # 3

## from example(warpbreaks)
require(stats)

fm1 <- lm(breaks ~ wool * tension, data = warpbreaks)
length(fm1$call) # 3, lm() and two arguments.
length(formula(fm1)) # 3, ~ lhs rhs
```

lengths

listis.atomic

lengths(x, use.names = TRUE)

x	listexpressionNULL
use.names	namesx

xxlength(x[[i]])ilength
lengths

integerlength(x) $2^{31} - 1$ use.namesx

lengths(x)sapply(x, length)*applylengthxlengths(x)

length

```

require(stats)
## summarize by month
l <- split(airquality$Ozone, airquality$Month)
avgOz <- lapply(l, mean, na.rm=TRUE)
## merge result
airquality$avgOz <- rep(unlist(avgOz, use.names=FALSE), lengths(l))
## but this is safer and cleaner, but can be slower
airquality$avgOz <- unsplit(avgOz, airquality$Month)

## should always be true, except when a length does not fit in 32 bits
stopifnot(identical(lengths(l), vapply(l, length, integer(1L))))

## empty lists are not a problem
x <- list()
stopifnot(identical(lengths(x), integer()))

## nor are "list-like" expressions:
lengths(expression(u, v, 1+ 0:9))

## and we should dispatch to length methods
f <- c(rep(1, 3), rep(2, 6), 3)
dates <- split(as.POSIXlt(Sys.time() + 1:10), f)
stopifnot(identical(lengths(dates), vapply(dates, length, integer(1L))))

```

levels

levels

```

levels(x)
levels(x) <- value

```

```

x
value          levels(x)NULLfactorx

```

factor

```

NAvalueNA
levels(x) <- valueattr(x, "levels") <- value

```

nlevelsrelevelreorder

```

## assign individual levels
x <- gl(2, 4, 8)
levels(x)[1] <- "low"
levels(x)[2] <- "high"
x

## or as a group
y <- gl(2, 4, 8)
levels(y) <- c("low", "high")
y

## combine some levels
z <- gl(3, 2, 12, labels = c("apple", "salad", "orange"))
z
levels(z) <- c("fruit", "veg", "fruit")
z

## same, using a named list
z <- gl(3, 2, 12, labels = c("apple", "salad", "orange"))
z
levels(z) <- list("fruit" = c("apple", "orange"),
                  "veg"   = "salad")
z

## we can add levels this way:
f <- factor(c("a", "b"))
levels(f) <- c("c", "a", "b")
f

f <- factor(c("a", "b"))
levels(f) <- list(C = "C", A = "a", B = "b")
f

```

```
libcurlVersion
```

```
libcurl
```

```
libcurlVersion()
```

```
libcurl""libcurl
```

```

ssl_version      "none"libcurl"SecureTransport"
libssh_version   libsshscpsftp"libssh2/1.5.0"
protocols

```

```
libcurllibcurllibcurl-featurelibcurlVersion
```

```
extSoftVersion
curlGetHeadersdownload.fileurllibcurl
https://curl.se/docs/sslcerts.htmlhttps://curl.se/docs/ssl-compared.html
libcurl

libcurlVersion()
```

libPaths

```
.libPaths

.libPaths(new, include.site = TRUE)

.Library
.Library.site

new          path.expand*?[Sys.glob
include.site .Library.siteTRUE.libPathsnew

.Librarylibrary
.Library.site
.libPathsnewunique(c(new,      .Library.site,      .Library))include.siteFALSEnew
.Library.sitenew
new
R_LIBSR_LIBS_USERR_LIBS_SITENULLR_LIBS_USER
.Library.siteR_LIBS_SITESite-libraryR_HOME.libPaths()R_LIBSR_LIBS_USERR_LIBS
R_LIBS_USERR/-library/Library/R///librarySys.info()["machine"]R/win-library/
LOCALAPPDATA
R_LIBS_USERR_LIBS_SITE%%%%
%V 2.5.0
%v 2.5
%p R.version$platform
%o R.version$os
%a R.version$arch
version%U%SR_LIBS_USERR_LIBS_SITE
.libPaths.Library.Library.site.Library.siteRprofile.site.libPaths(.libPaths())
normalizePath(winslash = "/")
LOCALAPPDATA C:\Users\username\AppData\Localshell.exec(Sys.getenv("LOCALAPPDATA"))
```

library

```
.libPaths()          # all library trees R knows about
```

library

libraryrequire

```
library(package, help, pos = 2, lib.loc = NULL,
         character.only = FALSE, logical.return = FALSE,
         warn.conflicts, quietly = FALSE,
         verbose = getOption("verbose"),
         mask.ok, exclude, include.only,
         attach.required = missing(include.only))
```

```
require(package, lib.loc = NULL, quietly = FALSE,
         warn.conflicts,
         character.only = FALSE,
         mask.ok, exclude, include.only,
         attach.required = missing(include.only))
```

```
conflictRules(pkg, mask.ok = NULL, exclude = NULL)
```

packagehelp	character.onlyFALSETRUE
pos	search()
lib.loc	NULLNULL. libPaths()
character.only	packagehelp
logical.return	TRUEFALSETRUE
warn.conflicts	TRUE conflicts TRUEFALSEconflicts.policy
verbose	TRUE
quietly	TRUE
pkg	
mask.ok	
excludeinclude.only	
	libraryrequire
attach.required	
	DependsDESCRIPTION


```
library(package)require(package)packagerequireFALSElibrary()detach(unload      =
TRUE)unloadNamespacerequireNamespace
suppressPackageStartupMessages
library(package)help(lib.loc"libraryIQR".packages(all = TRUE)installed.packages())
library(help = )"packageInfo"NULLlib.loc
```

```
libraryTRUEFALSElogical.returnTRUElibrary()"libraryIQR"library(help=)
"packageInfo"
require
```

```
conflicts.policywarn.conflictsTRUE"strict"mask.okexcludeinclude.onlylibrary
requiremask.okexcludeconflictRules
conflicts.policy"depends.ok"
conflicts.policy
error TRUE
warn FALSE
generics.ok TRUE
depends.ok TRUE
can.mask
```

```
getOption("checkPackageLicense") == TRUE~/.R/licensed
https://en.wikipedia.org/wiki/FOSSavailable.packages
R_HOME/etc/licensed.site
```

```
library.noGenerics
```

```
libraryrequireDESCRIPTIONBuilt:
R.version$platformi386-pc-linux-gnux86_64-unknown-linux-gnu
libraryrequireDESCRIPTION
```

```
.libPaths.packages
attachdetachsearchobjectsautoloadrequireNamespacelibrary.dynamdata
install.packagesinstalled.packagesINSTALLREMOVE
options(defaultPackages=)Startup
```

```

library()                # list all available packages
library(lib.loc = .Library) # list all packages in the default library
library(help = splines)    # documentation on package 'splines'
library(splines)          # attach package 'splines'
require(splines)          # the same
search()                  # "splines", too
detach("package:splines")

# if the package name is in a character vector, use
pkg <- "splines"
library(pkg, character.only = TRUE)
detach(pos = match(paste("package", pkg, sep = ":"), search()))

require(pkg, character.only = TRUE)
detach(pos = match(paste("package", pkg, sep = ":"), search()))

require(nonexistent)      # FALSE
## Not run:
## if you want to mask as little as possible, use
library(mypkg, pos = "package:base")

## End(Not run)

```

library.dynam

```

library.dynam(chname, package, lib.loc,
              verbose = getOption("verbose"),
              file.ext = .Platform$dynlib.ext, ...)

library.dynam.unload(chname, libpath,
                    verbose = getOption("verbose"),
                    file.ext = .Platform$dynlib.ext)

.dynLibs(new)

chname
package
lib.loc
libpath
verbose      options
file.ext     .
...          dyn.load
new          "DLLInfo"

```

```
dyn.load
```

```
library.dynam.onLoad.so.sl.dll
```

```
library.dynam.unload.onUnload.dynLibs()
```

```
.dynLibslibrary.dynam
```

```
chnamelibrary.dynam"DLLInfoList"
```

```
chname"DLLInfo""DLLInfo"$
```

```
library.dynam.unload"DLLInfo"
```

```
.dynLibs"DLLInfoList"
```

```
dyn.unloadlibrary.dynamlibrary.dynam.unload.dynLibslibrary.dynam
```

```
dyn.unload
```

```
getLoadedDLLs"DLLInfo""DLLInfoList"
```

```
.onLoadlibrarydyn.load.packages.libPaths
```

```
SHLIB
```

```
## Which DLLs were dynamically loaded by packages?
```

```
library.dynam()
```

```
## More on library.dynam.unload() :
```

```
require(nlme)
```

```
nlme:::onUnload # shows library.dynam.unload() call
```

```
detach("package:nlme") # by default, unload=FALSE , so,
```

```
tail(library.dynam(), 2)# nlme still there
```

```
## How to unload the DLL ?
```

```
## Best is to unload the namespace, unloadNamespace("nlme")
```

```
## If we need to do it separately which should be exceptional:
```

```
pd.file <- attr(packageDescription("nlme"), "file")
```

```
library.dynam.unload("nlme", libpath = sub("/Meta.*", '', pd.file))
```

```
tail(library.dynam(), 2)# 'nlme' is gone now
```

```
unloadNamespace("nlme") # now gives warning
```

license

```
license()
licence()
```

```
/doc/COPYINGRShowDoc("COPYING")RShowDoc("GPL-3")
/licenses/LGPL-2.1RShowDoc("LGPL-2.1")RShowDoc("LGPL-3")
```

list

```
list(...)
pairlist(...)
```

```
as.list(x, ...)
## S3 method for class 'environment'
as.list(x, all.names = FALSE, sorted = FALSE, ...)
as.pairlist(x)
```

```
is.list(x)
is.pairlist(x)
```

```
alist(...)
```

```
...
x
all.names
sorted          names
```

formals

```
listpairlistvaluetag = value
alistlistalistformals
as.listas.listas.vector(mode = "list")as.vectoras.listnamesas.character
is.listTRUElistpairlistlength> 0is.pairlistTRUENULL
"environment"as.listsorted = TRUEas.environment()
pairlist()NULLlist()
as.pairlistas.vector(x, "pairlist")as.vector
listis.listis.pairlist
```

```
vector("list", length)cformalsunlistas.list()
list
```

```
require(graphics)

# create a plotting structure
pts <- list(x = cars[,1], y = cars[,2])
plot(pts)

is.pairlist(.Options) # a user-level pairlist

## "pre-allocate" an empty list of length 5
vector("list", 5)

# Argument lists
f <- function() x
# Note the specification of a "..." argument:
formals(f) <- al <- alist(x = , y = 2+3, ... = )
f
al

## environment->list coercion

e1 <- new.env()
e1$a <- 10
e1$b <- 20
as.list(e1)
```

list.files

```
list.files(path = ".", pattern = NULL, all.files = FALSE,
           full.names = FALSE, recursive = FALSE,
           ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)

dir(path = ".", pattern = NULL, all.files = FALSE,
   full.names = FALSE, recursive = FALSE,
   ignore.case = FALSE, include.dirs = FALSE, no.. = FALSE)

list.dirs(path = ".", full.names = TRUE, recursive = TRUE)
```

```
path          getwd()path.expand
pattern
all.files     FALSETRUE
full.names    TRUEFALSE
recursive
ignore.case
include.dirs
no..          ". "" .."
```

```
full.names = TRUE
list.dirsall.files = TRUErecursive = TRUEpath
dirlist.files
```

```
file.infofile.accessfilesfile.choose
glob2rx
Sys.globbasenamedirname
```

```
list.files(R.home())
## Only files starting with a-l or r
## Note that a-l is locale-dependent, but using case-insensitive
## matching makes it unambiguous in English locales
dir("../..", pattern = "[a-lr]", full.names = TRUE, ignore.case = TRUE)

list.dirs(R.home("doc"))
list.dirs(R.home("doc"), full.names = FALSE)
```

list2DF

```
list2DF(x = list(), nrow = 0)
```

```
x
nrow      x
```

data.frame

```
## Create a data frame holding a list of character vectors and the
## corresponding lengths:
x <- list(character(), "A", c("B", "C"))
n <- lengths(x)
list2DF(list(x = x, n = n))

## Create data frames with no variables and the desired number of rows:
list2DF()
list2DF(nrow = 3L)
```

list2env

list xenvironmentx

```
list2env(x, envir = NULL, parent = parent.frame(),
        hash = (length(x) > 100), size = max(29L, length(x)))
```

```
x          listnames(x)""
envir      environmentNULL
parent     envir = NULLnew.env
hash       envir = NULLnew.env
size       envir = NULL, hash = TRUEnew.env
```

`environmentnew.env`
`envir`
`NULL`
`env`
`envir`

`environmentnew.env`
`as.environment`
`as.list.environment`

```
L <- list(a = 1, b = 2:4, p = pi, ff = gl(3, 4, labels = LETTERS[1:3]))
e <- list2env(L)
ls(e)
stopifnot(ls(e) == sort(names(L)),
          identical(L$b, e$b)) # "$" working for environments as for lists

## consistency, when we do the inverse:
ll <- as.list(e) # -> dispatching to the as.list.environment() method
rbind(names(L), names(ll)) # not in the same order, typically,
# but the same content:
stopifnot(identical(L [sort.list(names(L))],
                    ll[sort.list(names(ll))]))

## now add to e -- can be seen as a fast "multi-assign":
list2env(list(abc = LETTERS, note = "just an example",
             df = data.frame(x = rnorm(20), y = rbinom(20, 1, prob = 0.2))),
         envir = e)
utils::ls.str(e)
```

`load`

`save`

`load(file, envir = parent.frame(), verbose = FALSE)`

`file`
`envir`
`verbose`


```
loadsaveurl  
"rb"gzfilegzcongzcon
```

```
1971:1977RD[ABX]1  
verboseTRUEverbose
```

```
ascii = TRUEload  
load().GlobalEnvenvir = attach()load()search
```

```
savedownload.fileattachload()  
unserializereadRDS
```

```
## save all data  
xx <- pi # to ensure there is some data  
save(list = ls(all.names = TRUE), file= "all.rda")  
rm(xx)  
  
## restore the saved values to the current environment  
local({  
  load("all.rda")  
  ls()  
})  
  
xx <- exp(1:3)  
## restore the saved values to the user's workspace  
load("all.rda") ## which is here *equivalent* to  
## load("all.rda", .GlobalEnv)  
## This however annihilates all objects in .GlobalEnv with the same names !  
xx # no longer exp(1:3)  
rm(xx)  
attach("all.rda") # safer and will warn about masked objects w/ same name in .GlobalEnv  
ls(pos = 2)  
## also typically need to cleanup the search path:  
detach("file:all.rda")  
  
## clean up (the example):  
unlink("all.rda")  
  
## Not run:  
con <- url("http://some.where.net/R/data/example.rda")  
## print the value to see what objects were created.
```

```

print(load(con))
close(con) # url() always opens the connection

## End(Not run)

```

locales

```

Sys.getlocale (category = "LC_ALL")
Sys.setlocale (category = "LC_ALL", locale = "")
.LC.categories

```

```

category      "LC_ALL""LC_COLLATE""LC_CTYPE""LC_MONETARY""LC_NUMERIC"
               "LC_TIME""LC_MESSAGES""LC_PAPER""LC_MEASUREMENT".LC.categories
               Sys.getlocale(.)""LC_PAPER"
locale        ""

```

```

"C""POSIX""LC_CTYPE""LC_COLLATE"sort"LC_MONETARY"Sys.localeconv"LC_TIME"
as.POSIXltstrptime
"LC_MESSAGES""C"LANGUAGESys.setLanguage()Sys.setlocale
"LC_ALL""LC_COLLATE""LC_CTYPE""LC_MONETARY""LC_TIME"

```

```

Sys.setlocale("LC_CTYPE", )
LANGUAGE"LC_MESSAGES"
icuSetCollate"LC_COLLATE"Sys.setlocale

```

```

Sys.setlocaleNULL
category = "LC_ALL""/"";"foo <- Sys.getlocale()Sys.setlocale("LC_ALL", locale =
foo)

```

```

locale -alocale-genlocale -a
l10n_info()$codepagel10n_info()$system.codepage".UTF-8""C""

```

```

"LC_NUMERIC""C"save
options(OutDec)
"LC_NUMERIC"

```

```
Sys.getlocale("LC_CTYPE").UTF-8.utf8UTF-8ISO8859-15iso885915iso885915@euro
ISO8859-15@eurol10n_info
```

```
strptimecategory = "LC_TIME"Sys.localeconv
l10n_info
```

`Sys.setLanguage`

```
Sys.getlocale()

## Date-time related :
Sys.getlocale("LC_TIME") -> olcT
then <- as.POSIXlt("2001-01-01 01:01:01", tz = "UTC")
## Not run:
c(m = months(then), wd = weekdays(then)) # locale specific
Sys.setlocale("LC_TIME", "de") # Solaris: details are OS-dependent
Sys.setlocale("LC_TIME", "de_DE") # Many Unix-alikes
Sys.setlocale("LC_TIME", "de_DE.UTF-8") # Linux, macOS, other Unix-alikes
Sys.setlocale("LC_TIME", "de_DE.utf8") # some Linux versions
Sys.setlocale("LC_TIME", "German.UTF-8") # Windows
Sys.getlocale("LC_TIME") # the last one successfully set above
c(m = months(then), wd = weekdays(then)) # in C_TIME locale 'cT' ; typically German

## End(Not run)
Sys.setlocale("LC_TIME", "C")
c(m = months(then), wd = weekdays(then)) # "standard" (still platform specific ?)
Sys.setlocale("LC_TIME", olcT) # reset to previous

## Other locales
Sys.getlocale("LC_PAPER") # may or may not be set
.LC.categories # of length 9 on all platforms

## Not run: Sys.setlocale("LC_COLLATE", "C") # turn off locale-specific sorting,
# usually (but not on all platforms)
Sys.setenv("LANGUAGE" = "es") # set the language for error/warning messages

## End(Not run)
## some nice formatting; should work on most platforms,
## macOS does not name the entries.
sep <- switch(Sys.info()[["sysname"]],
  "Darwin" =, "SunOS" = "/",
  "Linux" =, "Windows" = ";")
##' named vector from a "full" Sys.getlocale() :
asNvec <- function(loc) {
  sl <- strsplit(strsplit(loc, sep)[[1L]], "=")
  if(all(lengths(sl) == 2L))
    setNames(sapply(sl, `[`, 2L), sapply(sl, `[`, 1L))
  else
    setNames(as.character(sl), .LC.categories[1+seq_along(sl)])
}
```

```
print.Dlist(lloc <- asNvec(Sys.getlocale()))
## R-supported ones (but LC_ALL):
lloc[.LC.categories[-1]]
```

log

loglog10log2log(x, base)base

log1p(x)log(1 + x) $|x| \ll 1$

exp

expm1(x) $\exp(x) - 1$ $|x| \ll 1$

log(x, base = exp(1))

logb(x, base = exp(1))

log10(x)

log2(x)

log1p(x)

exp(x)

expm1(x)

x

base e exp(1)

logbMath

log10log2loglog

logblogloglogb

log

xlog(0)-Inflog(x)xNaNexp(-Inf)0

$[-\pi, \pi]$

expexpm1loglog10log2log1pMath

logxbaseMathbaselog

log1pexpm1dlnrel<https://netlib.org/slatec/fnlib/dlnrel.f>log1p(y) = x

```
loglog10exp
logb
```

TrigsqrtArithmetic

```
log(exp(3))
log10(1e7) # = 7

x <- 10^-(1+2*1:9)
cbind(deparse.level=2, # to get nice column names
      x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

Logic

```
! x
x & y
x && y
x | y
x || y
xor(x, y)
```

```
isTRUE (x)
isFALSE(x)
```

```
xy          rawlogicaldoublenumericintegercomplex
```

```
!
&&&|||if
&&||
xor
isTRUE(x){ is.logical(x) && length(x) == 1 && !is.na(x) && x }isFALSE()
if(isTRUE(cond))if(cond)NA
isTRUE <- function(x) identical(x, TRUE)x <- c(val = TRUE)
!&|xor!
!&|OpsLogicOps
NAxyNANANA & TRUENANA & FALSEFALSE
```

```
!xxx
|&xorwarningtsp
||&&isTRUE
```

```
!&|Logice1, e2
```

```
`&`(x, y)Ops
```

```
TRUElogical
anyall
Syntax
L %||% R LNULLR
bitwAnd
```

```
y <- 1 + (x <- stats::rpois(50, lambda = 1.5) / 4 - 1)
x[(x > 0) & (x < 1)] # all x values between 0 and 1
if (any(x == 0) || any(y == 0)) "zero encountered"

## construct truth tables :

x <- c(NA, FALSE, TRUE)
names(x) <- as.character(x)
outer(x, x, `&`) ## AND table
outer(x, x, `|`) ## OR table
```

logical

```
"logical"
```

```
TRUE
FALSE
T; F

logical(length = 0)
as.logical(x, ...)
is.logical(x)
```

```
length
```

```
x
```

```
...
```

```
TRUEFALSETFlogical(1)
```

```
as.logical"logical"
```

```
TRUE1LFALSE0LNANA_integer_
```

```
logicalFALSE
```

```
as.logicalFALSETRUEfactorlevelsas.vectorc("T", "TRUE", "True", "true")c("F",  
"FALSE", "False", "false")NA
```

```
is.logicalTRUEFALSE
```

NALogic

```
## non-zero values are TRUE
```

```
as.logical(c(pi,0))
```

```
if (length(letters)) cat("26 is TRUE\n")
```

```
## logical interpretation of particular strings
```

```
charvec <- c("FALSE", "F", "False", "false", "fAlse", "0",  
"TRUE", "T", "True", "true", "tRue", "1")
```

```
as.logical(charvec)
```

```
## factors are converted via their levels, so string conversion is used
```

```
as.logical(factor(charvec))
```

```
as.logical(factor(c(0,1))) # "0" and "1" give NA
```

LongVectors

```
231
```

```
231 - 1
```

```
252
```

```
231 - 1
```

```
as.integer.C.Fortran.Call
```

```
 $m \times n$ %%crossprodsvdqrsolveeigen
```

```
chol
```

`lower.tri`

TRUE

```
lower.tri(x, diag = FALSE)
upper.tri(x, diag = FALSE)
```

```
x          length(dim(x)) == 2as.matrix(x)
diag
```

```
diagmatrixrowcollower.tri()upper.tri()
```

```
(m2 <- matrix(1:20, 4, 5))
lower.tri(m2)
m2[lower.tri(m2)] <- NA
m2
```

`ls`

`lsobjects``ls``ls``browser`

```
ls(name, pos = -1L, envir = as.environment(pos),
   all.names = FALSE, pattern, sorted = TRUE)
objects(name, pos = -1L, envir = as.environment(pos),
        all.names = FALSE, pattern, sorted = TRUE)
```

<code>name</code>	<code>name</code>
<code>pos</code>	<code>name</code>
<code>envir</code>	<code>name</code>
<code>all.names</code>	<code>TRUEFALSE.</code>
<code>pattern</code>	<code>patternglob2rx</code>
<code>sorted</code>	<code>characterls()</code>

```
namesearchenvironmentsys.frame
```

`lsobjects``pos``envir`
`sorted = TRUE``Sys.getlocalesorted = FALSE`

[glob2rx](#)

[ls.strstraproposfindgrepclassmethods](#)

```
.Ob <- 1
ls(pattern = "0")
ls(pattern= "0", all.names = TRUE)    # also shows ".[foo]"

# shows an empty list because inside myfunc no variables are defined
myfunc <- function() {ls()}
myfunc()

# define a local variable inside myfunc
myfunc <- function() {y <- 1; ls()}
myfunc()          # shows "y"
```

`make.names`

```
make.names(names, unique = FALSE, allow_ = TRUE)
```

```
names
unique      TRUE
allow_
```

```
".2way"
```

```
"X"."NA"make.unique
```

```
names
```

```
allow_ = FALSE
```

```
allow_ = FALSE
```

[make.uniquenamescharacterdata.frame](#)

```
make.names(c("a and b", "a-and-b"), unique = TRUE)
# "a.and.b" "a.and.b.1"
make.names(c("a and b", "a_and_b"), unique = TRUE)
# "a.and.b" "a_and_b"
make.names(c("a and b", "a_and_b"), unique = TRUE, allow_ = FALSE)
# "a.and.b" "a.and.b.1"
make.names(c("", "X"), unique = TRUE)
# "X.1" "X" currently; R up to 3.0.2 gave "X" "X.1"

state.name[make.names(state.name) != state.name] # those 10 with a space
```

[make.unique](#)

```
make.unique(names, sep = ".")
```

names

sep

```
make.uniquemake.unique(c(A, B)) == make.unique(c(make.unique(A), B))
```

```
make.unique
```

```
Amake.unique(c(A, B))A
```

names

[make.names](#)

```

make.unique(c("a", "a", "a"))
make.unique(c(make.unique(c("a", "a")), "a"))

make.unique(c("a", "a", "a.2", "a"))
make.unique(c(make.unique(c("a", "a")), "a.2", "a"))

## Now show a bit where this is used :
trace(make.unique)
## Applied in data.frame() constructions:
(d1 <- data.frame(x = 1, x = 2, x = 3)) # direct
d2 <- data.frame(data.frame(x = 1, x = 2), x = 3) # pairwise
stopifnot(identical(d1, d2),
           colnames(d1) == c("x", "x.1", "x.2"))
untrace(make.unique)

```

mapply

```

mapplysapplymapplyFUN
.mapply()mapply()

```

```

mapply(FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE,
        USE.NAMES = TRUE)
.mapply(FUN, dots, MoreArgs)

```

```

FUN          match.fun
...
dots         listpairlist...
MoreArgs     FUN
SIMPLIFY     simplifysapply
USE.NAMES

```

```

mapplyFUN...list()MoreArgs...MoreArgs
...dots[length

```

```
listSIMPLIFY = TRUE

```

```

sapplymapply()
outer

```

```

mapply(rep, 1:4, 4:1)

mapply(rep, times = 1:4, x = 4:1)

mapply(rep, times = 1:4, MoreArgs = list(x = 42))

mapply(function(x, y) seq_len(x) + y,
        c(a = 1, b = 2, c = 3), # names from first
        c(A = 10, B = 0, C = -10))

word <- function(C, k) paste(rep.int(C, k), collapse = "")
## names from the first, too:
utils::str(L <- mapply(word, LETTERS[1:6], 6:1, SIMPLIFY = FALSE))

mapply(word, "A", integer()) # gave Error, now list()

```

marginSums

```

marginSums(x, margin = NULL)
margin.table(x, margin = NULL)

x          table
margin     12c(1, 2)xdimnames

```

```
marginxmargin
```

```
margin.table
```

```

rowSumscolSums
proportionsaddmargins

```

```

m <- matrix(1:4, 2)
marginSums(m, 1) # = rowSums(m)
marginSums(m, 2) # = colSums(m)

DF <- as.data.frame(UCBAdmissions)
tbl <- xtabs(Freq ~ Gender + Admit, DF)
tbl
marginSums(tbl, "Gender") # a 1-dim "table"
rowSums(tbl)             # a numeric vector

```

mat.or.vec

mat.or.vecnrncncnrnc

mat.or.vec(nr, nc)

nrnc

mat.or.vec(3, 1)
mat.or.vec(3, 2)

match

match
%in%

match(x, table, nomatch = NA_integer_, incomparables = NULL)

x %in% table

x	NULL
table	NULL
nomatch	integer
incomparables	xnomatchFALSENULL

%in%
"%in%" <- function(x, table) match(x, table, nomatch = 0) > 0
[mtfrm](#)xtableincomparables

NANANaNNaNNaXNA
"bytes"[Encoding](#)
%in%NAif

x
matchtablenomatch
x[i]table[j]ijjnomatch
%in%xTRUEFALSENA

[pmatch](#)[charmatch](#)[match.arg](#)[findInterval](#)
[is.element](#)[%in%](#)
[unique](#)[duplicated](#)[match\(\)](#)[==](#)[NaN](#)[NaN](#)

```
## The intersection of two sets can be defined via match():
## Simple version:
## intersect <- function(x, y) y[match(x, y, nomatch = 0)]
intersect # the R function in base is slightly more careful
intersect(1:10, 7:20)

1:10 %in% c(1,3,5,9)
sstr <- c("c","ab","B","bba","c",NA,"e","bla","a","Ba","%")
sstr[sstr %in% c(letters, LETTERS)]

"%w/o%" <- function(x, y) x[!x %in% y] #-- x without y
(1:10) %w/o% c(3,7,12)
## Note that setdiff() is very similar and typically makes more sense:
      c(1:6,7:2) %w/o% c(3,7,12) # -> keeps duplicates
setdiff(c(1:6,7:2),      c(3,7,12)) # -> unique values

## Illuminating example about NA matching
r <- c(1, NA, NaN)
zN <- c(complex(real = NA , imaginary = r ), complex(real = r , imaginary = NA ),
      complex(real = r , imaginary = NaN), complex(real = NaN, imaginary = r ))
zM <- cbind(Re=Re(zN), Im=Im(zN), match = match(zN, zN))
rownames(zM) <- format(zN)
zM ##--> many "NA's" (= 1) and the four non-NA's (3 different ones, at 7,9,10)

length(zN) # 12
unique(zN) # the "NA" and the 3 different non-NA NaN's
stopifnot(identical(unique(zN), zN[c(1, 7,9,10)]))

## very strict equality would have 4 duplicates (of 12):
symnum(outer(zN, zN, Vectorize(identical,c("x","y"))),
      FALSE,FALSE,FALSE,FALSE))
## removing "(very strictly) duplicates",
i <- c(5,8,11,12) # we get 8 pairwise non-identicals :
Ixy <- outer(zN[-i], zN[-i], Vectorize(identical,c("x","y"))),
      FALSE,FALSE,FALSE,FALSE)
stopifnot(identical(Ixy, diag(8) == 1))
```

[match.arg](#)

[match.arg](#)[choices](#)

[match.arg](#)(arg, choices, several.ok = FALSE)

```
arg          several.okTRUENULLchoices[1]
```

```
choices
```

```
several.ok   arg
```

```
match.arg(arg) argmatch.argargchoicesseveral.okTRUE
```

```
pmatcharg""pmatch
```

```
several.okseveral.okarg
```

```
pmatchmatch.funmatch.call
```

```
require(stats)
## Extends the example for 'switch'
center <- function(x, type = c("mean", "median", "trimmed")) {
  type <- match.arg(type)
  switch(type,
    mean = mean(x),
    median = median(x),
    trimmed = mean(x, trim = .1))
}
x <- rcauchy(10)
center(x, "t")      # Works
center(x, "med")    # Works
try(center(x, "m")) # Error
stopifnot(identical(center(x),      center(x, "mean")),
  identical(center(x, NULL), center(x, "mean")) )

## Allowing more than one 'arg' and hence more than one match:
match.arg(c("gauss", "rect", "ep"),
  c("gaussian", "epanechnikov", "rectangular", "triangular"),
  several.ok = TRUE)
match.arg(c("a", ""), c("", NA, "bb", "abc"), several.ok=TRUE) # |--> "abc"
```

```
match.call
```

```
match.call
```

```
match.call(definition = sys.function(sys.parent()),
  call = sys.call(sys.parent()),
  expand.dots = TRUE,
  envir = parent.frame(2L))
```

```
definition      match.call
call            definitioncall
expand.dots     .....
envir          ...call
```

```
match.call
match.call

      callexpand.dots = TRUE
      model.frameexpand.dots = FALSE...lm
match.calldefinition

call
```

[sys.call\(\)](#)[callpmatchmatch.argmatch.fun](#)

```
match.call(get, call("get", "abc", i = FALSE, p = 3))
## -> get(x = "abc", pos = 3, inherits = FALSE)
fun <- function(x, lower = 0, upper = 1) {
  structure((x - lower) / (upper - lower), CALL = match.call())
}
fun(4 * atan(1), u = pi)
```

match.fun

```
match.fun(FUN, descend = TRUE)
```

```
FUN
descend
```

```
match.fun
FUNgetsubstitute
descend = TRUEmatch.funFUN
applylapplyoutersweep
```


FUN

descend

attach

[match.argget](#)

```
# Same as get("*"):
match.fun("*")
# Overwrite outer with a vector
outer <- 1:5
try(match.fun(outer, descend = FALSE)) #-> Error: not a function
match.fun(outer) # finds it anyway
is.function(match.fun("outer")) # as well
```

MathFun

$\text{abs}(x)\text{sqrt}(x)\sqrt{x}$

$\text{abs}(x)$
 $\text{sqrt}(x)$

x [complex](#)

[Mathzabs\(z\) == Mod\(z\)sqrt\(z\) == z^0.5](#)
 $\text{abs}(x)$ [integer](#)[xinteger](#)[logical](#)

[Math](#)

[ArithmeticlogsinSpecial](#)

[sqrt](#)

```
require(stats) # for spline
require(graphics)
xx <- -9:9
plot(xx, sqrt(abs(xx)), col = "red")
lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

matmult

`x %*% y`

`xy`

`as.matrix`

`xy`

`matOpsxy`

`drop`

`options("matprod")`

`crossprod()``tcrossprod()``matrixArithmeticdiag`

```
x <- 1:4
(z <- x %*% x)    # scalar ("inner") product (1 x 1 matrix)
drop(z)          # as scalar

y <- diag(x)
z <- matrix(1:12, ncol = 3, nrow = 4)
y %*% z
y %*% x
x %*% z
```

matrix

matrix

as.matrix

is.matrix

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,  
       dimnames = NULL)
```

```
as.matrix(x, ...)
```

```
## S3 method for class 'data.frame'
```

```
as.matrix(x, rownames.force = NA, ...)
```

```
is.matrix(x)
```

data [expressionas.vector](#)

nrow

ncol

byrow FALSE

dimnames [dimnames](#)NULLlistNULL

x

...

rownames.force NULL[rownames](#)NANULL

nrowncoldata

datadatadataNA0NULL

is.matrixTRUEx"[dim](#)"FALSE[data.frame](#)

as.matrix[as.vector](#)[format](#)

as.matrixas.vector(x)

is.matrix

printinteger,7

```
dim(x) <- c(nx, ny)
```

```
dimnames(x) <- list(row_names, col_names)
```

[xclass\(x\)](#)[Date](#)

`data.matrix`

`arrayinherits(m, "array")matrixm`

```
is.matrix(as.matrix(1:10))
!is.matrix(warpbreaks) # data.frame, NOT matrix!
warpbreaks[1:10,]
as.matrix(warpbreaks[1:10,]) # using as.matrix.data.frame(.) method

## Example of setting row and column names
mdat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3, byrow = TRUE,
               dimnames = list(c("row1", "row2"),
                               c("C.1", "C.2", "C.3")))
mdat
```

`maxCol`

`max.col(m, ties.method = c("random", "first", "last"))`

`m`

`ties.method` `"random"`

`ties.method = "random"10-5`

`ties.method = "first"max.colunname(apply(m, 1, which.max))m`

`ties.method = "last"`

`nrow(m)`

`which.max`

```

table(mc <- max.col(swiss)) # mostly "1" and "5", 5 x "2" and once "4"
swiss[unique(print(mr <- max.col(t(swiss)))) , ] # 3 33 45 45 33 6

set.seed(1) # reproducible example:
(mm <- rbind(x = round(2*stats::runif(12)),
             y = round(5*stats::runif(12)),
             z = round(8*stats::runif(12))))
## Not run:
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x    1    1    1    2    0    2    2    1    1    0    0    0
y    3    2    4    2    4    5    2    4    5    1    3    1
z    2    3    0    3    7    3    4    5    4    1    7    5

## End(Not run)
## column indices of all row maxima :
utils::str(lapply(1:3, function(i) which(mm[i,] == max(mm[i,]))))
max.col(mm) ; max.col(mm) # "random"
max.col(mm, "first") # -> 4 6 5
max.col(mm, "last") # -> 7 9 11

```

mean

```

mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)

x          trim = 0
trim       x
na.rm      TRUEFALSENA
...

trimxxNA_real_
trimtrim

```

[weighted.meanmean.POSIXctcolMeans](#)

```

x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))

```

memCompress

```
memCompress(from, type = c("gzip", "bzip2", "xz", "zstd", "none"))
```

```
memDecompress(from,  
               type = c("unknown", "gzip", "bzip2", "xz", "zstd",  
                       "none"), asChar = FALSE)
```

```
from          memCompress"\n""bzip2"
```

```
type
```

```
asChar         $2^{31} - 1$ 
```

```
type = "none" type
```

```
type = "unknown" bzip2 type = "none"
```

```
gzip6
```

```
bzip2"BZh" $2^{31} - 1$ bzip2 -9
```

```
zstd
```

```
type = "xz"xz -9exz lzma
```

```
"gzip""bzip2"memCompress"xz"
```

```
asChar = TRUE
```

```
libdeflate
```

```
libdeflate type = "gzip"
```

```
https://github.com/ebiggers/libdeflate
```

```
extSoftVersion\(\)
```

```
extSoftVersionzliblibdeflatebzip2xz
```

```
https://en.wikipedia.org/wiki/Data\_compressionhttps://zlib.net/

```
wikipedia.org/wiki/Gziphttp://www.bzip.org/https://en.wikipedia.org/wiki/Bzip2
```


```

```
https://en.wikipedia.org/wiki/XZ\_Uutils
```

```

txt <- readLines(file.path(R.home("doc"), "COPYING"))
sum(nchar(txt))
txt.gz <- memCompress(txt, "g") # "gzip", the default
length(txt.gz)
txt2 <- strsplit(memDecompress(txt.gz, "g", asChar = TRUE), "\n")[[1]]
stopifnot(identical(txt, txt2))
## as from R 4.4.0 this is detected if not specified.
txt2b <- strsplit(memDecompress(txt.gz, asChar = TRUE), "\n")[[1]]
stopifnot(identical(txt2b, txt2))

txt.bz2 <- memCompress(txt, "b")
length(txt.bz2)
## can auto-detect bzip2:
txt3 <- strsplit(memDecompress(txt.bz2, asChar = TRUE), "\n")[[1]]
stopifnot(identical(txt, txt3))

## xz compression is only worthwhile for large objects
txt.xz <- memCompress(txt, "x")
length(txt.xz)
txt3 <- strsplit(memDecompress(txt.xz, asChar = TRUE), "\n")[[1]]
stopifnot(identical(txt, txt3))

## test decompressing a gzip-ed file
tf <- tempfile(fileext = ".gz")
con <- gzfile(tf, "w")
writelines(txt, con)
close(con)
(nf <- file.size(tf))
# if (nzchar(Sys.which("file"))) system2("file", tf)
foo <- readBin(tf, "raw", n = nf)
unlink(tf)
## will detect the gzip header and choose type = "gzip"
txt3 <- strsplit(memDecompress(foo, asChar = TRUE), "\n")[[1]]
stopifnot(identical(txt, txt3))

```

memlimits

```

mem.maxVSize(vsize = 0)
mem.maxNSize(nsize = 0)

```

```

vsize
nsize

```

Inf

[Memory](#)

mem.maxVSizeInf

Memory

Memory

--min-nsiz--min-vsizeR_NSIZER_VSIZER_MAX_VSIZE
R_GC_MEM_GROW
`gc()``gcinfo`(TRUE)
--max-ppsize

Memory-limits

`gcobject.size(a)``memory.profile`

Memory-limits

Memory

cannot allocate vector of size
cannot allocate vector of length $2^{31} - 1 \approx 2 \cdot 10^9$


```
limitulimitbash
```

```
ulimit -t 600 -v 4000000
```

```
csch
```

```
limit cputime 10m
```

```
limit vmemoryuse 4096m
```

<https://learn.microsoft.com/en-gb/windows/desktop/Memory/physical-address-extension><https://learn.microsoft.com/en-gb/windows/desktop/Memory/4-gigabyte-tuning>

```
object.size(a)a
```

```
memory.profile
```

```
SEXPREC
```

```
memory.profile()
```

```
Rinternals.h
```

```
typeof
```

```
gcRprofmemtracemem
```

```
memory.profile()
```

merge

```
merge(x, y, ...)

## Default S3 method:
merge(x, y, ...)

## S3 method for class 'data.frame'
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x", ".y"), no.dups = TRUE,
      incomparables = NULL, ...)

xy
byby.xby.y
all          all = Lall.x = Lall.y = LLTRUEFALSE
all.x        TRUExyNAyFALSExy
all.y        all.x
sort         by
suffixes     by
no.dups      suffixes
incomparables match
...

merge"data.frame"
by.xby.ymatch
"row.names"0
byby.xby.yNULLrxydim(r) = c(nrow(x)*nrow(y), ncol(x) + ncol(y))
all.xxNAyall.y
suffixes".x"".y"
by.xyno.dups

all = FALSEall.x = TRUEall.y = TRUEall = TRUENULLincomparables = NA

sort = FALSExyRow.names
```

[data.frame](#)[by](#)[bind](#)

[dendrogram](#)[merge](#)

```
authors <- data.frame(
  surname = c("Tukey", "Venables", "Tierney", "Ripley", "McNeil"),
  nationality = c("US", "Australia", "US", "UK", "Australia"),
  deceased = c("yes", rep("no", 4)))
authorN <- within(authors, { name <- surname; rm(surname) })
books <- data.frame(
  name = c("Tukey", "Venables", "Tierney",
           "Ripley", "Ripley", "McNeil", "R Core"),
  title = c("Exploratory Data Analysis",
            "Modern Applied Statistics ...",
            "LISP-STAT",
            "Spatial Statistics", "Stochastic Simulation",
            "Interactive Data Analysis",
            "An Introduction to R"),
  other.author = c(NA, "Ripley", NA, NA, NA, NA,
                  "Venables & Smith"))

(m0 <- merge(authorN, books))
(m1 <- merge(authors, books, by.x = "surname", by.y = "name"))
m2 <- merge(books, authors, by.x = "name", by.y = "surname")
stopifnot(exprs = {
  identical(m0, m2[, names(m0)])
  as.character(m1[, 1]) == as.character(m2[, 1])
  all.equal(m1[, -1], m2[, -1][ names(m1)[-1] ])
  identical(dim(merge(m1, m2, by = NULL)),
            c(nrow(m1)*nrow(m2), ncol(m1)+ncol(m2)))
})

## "R core" is missing from authors and appears only here :
merge(authors, books, by.x = "surname", by.y = "name", all = TRUE)

## example of using 'incomparables'
x <- data.frame(k1 = c(NA,NA,3,4,5), k2 = c(1,NA,NA,4,5), data = 1:5)
y <- data.frame(k1 = c(NA,2,NA,4,5), k2 = c(NA,NA,3,4,5), data = 1:5)
merge(x, y, by = c("k1","k2")) # NA's match
merge(x, y, by = "k1") # NA's match, so 6 rows
merge(x, y, by = "k2", incomparables = NA) # 2 rows
```

message

```
message(..., domain = NULL, appendLF = TRUE)
suppressMessages(expr, classes = "message")

packageStartupMessage(..., domain = NULL, appendLF = TRUE)
suppressPackageStartupMessages(expr)

.makeMessage(..., domain = NULL, appendLF = FALSE)
```

```
...           message
domain        gettextNAstop
appendLF
expr
classes
```

```
messagesstderr()
message
muffleMessage
suppressMessages
packageStartupMessagessuppressPackageStartupMessagessuppressMessages
.makeMessagemessagewarningstop...gettext
```

```
warningstopconditions
gettext
```

```
message("ABC", "DEF")
suppressMessages(message("ABC"))

testit <- function() {
  message("testing package startup messages")
  packageStartupMessage("initializing ...", appendLF = FALSE)
  Sys.sleep(1)
  packageStartupMessage(" done")
}

testit()
suppressPackageStartupMessages(testit())
suppressMessages(testit())
```

missing

missing

missing(x)

x

missing(x)xx <- match.arg(x)

missinglocal

[substituteNA](#)

```
myplot <- function(x, y) {  
  if(missing(y)) {  
    y <- x  
    x <- 1:length(y)  
  }  
  plot(x, y)  
}
```

mode

```
mode(x)  
mode(x) <- value  
storage.mode(x)  
storage.mode(x) <- value
```

x

value

```

modestorage.modetypeof(x)

mode(x) <- "newmode"modexnewmodeas.newmode"logical""integer""double""complex"
"raw""character""list""expression""name""symbol""function"

storage.mode(x) <- "newmode"mode<-"newmode"typeof"single"

"single"modestorage.modeattr(object, "Csingle")mode<-"single""double""Csingle"
TRUE

call"("

```

typeof

```

"integer""double""numeric"

"special""builtin""closure""function"

"symbol""name"

"language""("call"

```

typeoftype.convertattributes

```

require(stats)

sapply(options(), mode)

cex3 <- c("NULL", "1", "1:1", "1i", "list(1)", "data.frame(x = 1)",
  "pairlist(pi)", "c", "lm", "formals(lm)[[1]]", "formals(lm)[[2]]",
  "y ~ x", "expression((1))[[1]]", "(y ~ x)[[1]]",
  "expression(x <- pi)[[1]][[1]]")
lex3 <- sapply(cex3, function(x) eval(str2lang(x)))
mex3 <- t(sapply(lex3,
  function(x) c(typeof(x), storage.mode(x), mode(x))))
dimnames(mex3) <- list(cex3, c("typeof(.)", "storage.mode(.)", "mode(.)))
mex3

## This also makes a local copy of 'pi':
storage.mode(pi) <- "complex"
storage.mode(pi)
rm(pi)

```

mtfrm

```
match()-> "mtfrm"default"POSIXct""POSIXlt"
```

```
mtfrm(x)
```

```
x
```

```
matchmtfrmis.objectas.character
```

```
mtfrmmatchduplicatedunique==!=
```

```
x
```

NA

```
NANANA_integer_NA_real_NA_complex_NA_character_
```

```
is.na
```

```
is.na<-NA
```

```
anyNAany(is.na(x))
```

```
NA
```

```
is.na(x)
```

```
anyNA(x, recursive = FALSE)
```

```
## S3 method for class 'data.frame'
```

```
is.na(x)
```

```
is.na(x) <- value
```

```
x          is.naanyNANULL
```

```
recursive  anyNA
```

```
value      x
```

```
NA"NA"NA_character_"NA"NAis.na<-
```

```
is.naanyNA
```

```
is.na<-
```

```
NANANaNNaNaNaN
```

```
NATRUE/FALSETRUEFALSENA
```

```
anyNANULLany(is.na(x))recursive = FALSE
```

```
is.na(x)TRUENaNFALSENaNdimdimnamesnames
```

```
is.naNaNis.na
anyNA(recursive = FALSE)is.naanyNA(recursive = TRUE)anyNA
is.na
anyNA(NULL)is.na(NULL)logical(0)
```

[NaNis.nancomplete.cases](#)

[na.actionna.omitna.fail](#)

```
is.na(c(1, NA))      #> FALSE  TRUE
is.na(paste(c(1, NA))) #> FALSE FALSE
```

```
(xx <- c(0:4))
is.na(xx) <- c(2, 4)
xx          #> 0 NA  2 NA  4
anyNA(xx) # TRUE
```

```
# Some logical operations do not return NA
c(TRUE, FALSE) & NA
c(TRUE, FALSE) | NA
```

```
## Measure speed difference in a favourable case:
## the difference depends on the platform, on most ca 3x.
x <- 1:10000; x[5000] <- NaN # coerces x to be double
if(require("microbenchmark")) { # does not work reliably on all platforms
  print(microbenchmark(any(is.na(x)), anyNA(x)))
} else {
  nSim <- 2^13
  print(rbind(is.na = system.time(replicate(nSim, any(is.na(x)))),
              anyNA = system.time(replicate(nSim, anyNA(x)))))
}
```

```
## anyNA() can work recursively with list():
LL <- list(1:5, c(NA, 5:8), c("A","NA"), c("a", NA_character_))
L2 <- LL[c(1,3)]
sapply(LL, anyNA); c(anyNA(LL), anyNA(LL, TRUE))
sapply(L2, anyNA); c(anyNA(L2), anyNA(L2, TRUE))
```

```
## ... lists, and hence data frames, too:
dN <- dd <- USJudgeRatings; dN[3,6] <- NA
anyNA(dd) # FALSE
anyNA(dN) # TRUE
```

name

```
as.nameas.symbol
is.symbolis.nameTRUEFALSE
```

```
as.symbol(x)
is.symbol(x)
```

```
as.name(x)
is.name(x)
```

```
x
```

```
as.nameas.character""NA_character_`NA`
as.nameas.vector(x, "symbol")as.vector
is.nameis.symbol
```

```
as.nameas.symbol"symbol"typeof
is.nameis.symbolTRUEFALSE
```

```
callis.languagetypeof
plotmath
```

```
an <- as.name("arrg")
is.name(an) # TRUE
mode(an)    # name
typeof(an)  # symbol
```

names

```
names(x)
names(x) <- value
```

```
x
value          xNULL
```

```
namesnames<-"names"
environmentenvnames(env)names(as.list(env, all.names = TRUE))ls(env, all.names
= TRUE, sorted = FALSE)names(env)
valuexNAx
z <- "names<-"(z, "[<-"(names(z), 3, "c2"))
""""
NA
```

```
namesNULLxNULL
names<-names(x) <- valuevalue
```

```
namesdimnames[[1]]
slotNames()names()
```

```
slotNamesdimnames
```

```

# print the names attribute of the islands data set
names(islands)

# remove the names attribute
names(islands) <- NULL
islands
rm(islands) # remove the copy made

z <- list(a = 1, b = "c", c = 1:3)
names(z)
# change just the name of the third element.
names(z)[3] <- "c2"
z

z <- 1:3
names(z)
## assign just one name
names(z)[2] <- "b"
z

```

nargs

nargs

nargs()

foo(x,,z)x[]`[.some_method`(x,)

[argsformalssys.call](#)

```

tst <- function(a, b = 3, ...) {nargs()}
tst() # 0
tst(clicketyclack) # 1 (even non-existing)
tst(c1, a2, rr3) # 3

foo <- function(x, y, z, w) {
  cat("call was ", deparse(match.call()), "\n", sep = "")
  nargs()
}

```

```
foo()      # 0
foo(, , 3) # 3
foo(z = 3) # 1, even though this is the same call

nargs()   # not really meaningful
```

nchar

ncharx

nzchar

```
nchar(x, type = "chars", allowNA = FALSE, keepNA = NA)
```

```
nzchar(x, keepNA = FALSE)
```

x

```
type      c("bytes", "chars", "width")
```

```
allowNA    NA"bytes"
```

```
keepNA     NAxNA nchar() 2nzchar() TRUE nchar() NA keepNA = TRUE type "width"
```

type

bytes

chars

width [catch](#)chars

typekeepNA

[as.character](#)x[deparse](#)

```
ncharNA NA\_character\_ nchar() NA\_integer\_ keepNA 2
```

```
type = "width"
```

```
allowNA = TRUE NA!is.na(nchar(x, "chars", TRUE))
```

```
"bytes" Encoding NA allowNA = TRUE
```

```
nzcharxNA nzchar() keepNA NA NA
```

```
print() encodeString<U+2642>
```

```
"y\u306""\u200b"
```

nchar

<https://www.unicode.org/reports/tr11/>

strwidthpastesubstrstrsplit

```
x <- c("asfef", "qwerty", "yuiop[", "b", "stuff.blah.yech")
nchar(x)
# 5 6 6 1 15

nchar(deparse(mean))
# 18 17 <-- unless mean differs from base::mean

## NA behaviour as function of keepNA=* :
logi <- setNames(c(FALSE, NA, TRUE))
sapply(logi, \(k) data.frame(nchar = nchar(NA, keepNA=k),
                             nzchar = nzchar(NA, keepNA=k)))

x[3] <- NA; x
nchar(x, keepNA= TRUE) # 5 6 NA 1 15
nchar(x, keepNA=FALSE) # 5 6 2 1 15
stopifnot(identical(nchar(x), nchar(x, keepNA= TRUE)),
           identical(nchar(x, "w"), nchar(x, keepNA=FALSE)),
           identical(is.na(x), is.na(nchar(x))))

##' nchar() for all three types :
nchars <- function(x, ...)
  vapply(c("chars", "bytes", "width"),
        function(tp) nchar(x, tp, ...), integer(length(x)))

nchars("\u200b") # in R versions (>= 2015-09-xx):
## chars bytes width
##    1    3    0

data.frame(x, nchars(x)) ## all three types : same unless for NA
## force the same by forcing 'keepNA':
(ncT <- nchars(x, keepNA = TRUE)) ## .... NA NA NA ....
(ncF <- nchars(x, keepNA = FALSE))## .... 2 2 2 ....
stopifnot(apply(ncT, 1, function(.) length(unique(.))) == 1,
          apply(ncF, 1, function(.) length(unique(.))) == 1)
```

nlevels

nlevels(x)

x

[levels](#)

[levels\(x\)x](#)

[levelsfactor](#)

`nlevels(gl(3, 7)) # = 3`

[noquote](#)

`noquote(obj, right = FALSE)`

```
## S3 method for class 'noquote'
print(x, quote = FALSE, right = FALSE, ...)
```

```
## S3 method for class 'noquote'
c(..., recursive = FALSE)
```

<code>obj</code>	character
<code>right</code>	logical <code>print()</code> print.default()
<code>x</code>	<code>"noquote"</code>
<code>quote...</code>	print
<code>recursive</code>	c

```
noquote"noquote"c()".noquote"print.noquote"..."....
rightprint(x, right=*)rightxx <- noquote(*, right=TRUE)
class
```

`<maechler@stat.math.ethz.ch>`

[methodsclassprint](#)

```

letters
nql <- noquote(letters)
nql
nql[1:4] <- "oh"
nql[1:12]

cmp.logical <- function(log.v)
{
  ## Purpose: compact printing of logicals
  log.v <- as.logical(log.v)
  noquote(if(length(log.v) == 0)"" else c(".", "|")[1 + log.v])
}
cmp.logical(stats::runif(20) > 0.8)

chmat <- as.matrix(format(stackloss)) # a "typical" character matrix
## noquote(*, right=TRUE) so it prints exactly like a data frame
chmat <- noquote(chmat, right = TRUE)
chmat

```

norm

```
x"O""I""F""M""2"type
```

```
norm(x, type = c("O", "I", "F", "M", "2"))
```

```

x          norm()
type
          "O""o""1"
          "I""i"
          "F""f""E""e" x
          "M""m" x
          "2" svdx
          "O"type[1]

```

```

norm()dlange
"M"

```

```

norm = "2"DLANGE
https://netlib.org/lapack/

```

[rcond](#)

```
(x1 <- cbind(1, 1:10))
norm(x1)
norm(x1, "I")
norm(x1, "M")
stopifnot(all.equal(norm(x1, "F"),
                     sqrt(sum(x1^2))))

hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
h9 <- hilbert(9)
## all 5 (4 different) types of norm:
(nTyp <- eval(formals(base::norm)$type))
sapply(nTyp, norm, x = h9)
stopifnot(exprs = { # 0-extent matrices:
  sapply(nTyp, norm, x = matrix(, 1,0)) == 0
  sapply(nTyp, norm, x = matrix(, 0,0)) == 0
})
```

`normalizePath`

`normalizePath(path, winslash = "\\ ", mustWork = NA)`

`path`

`winslash` `c("/", "\\ ")`

`mustWork` `TRUE`
`NA`

[path.expandpaths](#)

`../realpathrealpath`

`winslashGetFinalPathNameByHandleGetFullPathNameGetLongPathName`
`GetFinalPathNameByHandleGetLongPathName`

`mustWork` = `FALSE`


```
mustWork = TRUE
```

```
/tmp/var/proc/fd/63path
```

```
cat(normalizePath(c(R.home(), tempdir())), sep = "\n")
```

NotYet

```
.NotYetImplemented()  
.NotYetUsed(arg, error = TRUE)
```

```
arg  
error          TRUEFALSE
```

[DeprecatedDefunct](#)

```
require(graphics)  
barplot(1:5, inside = TRUE) # 'inside' is not yet used
```

nrow

nrowncolxNCOLNROWas.matrix()cbind()

nrow(x)
ncol(x)
NCOL(x)
NROW(x)

x NULL

integerNULLncolnrow

ncolnrow

dimlengthdim()NULLnrow()ncol()NULLarraymatrix

```
ma <- matrix(1:12, 3, 4)
nrow(ma)  # 3
ncol(ma)  # 4
```

```
ncol(array(1:24, dim = 2:4)) # 3, the second dimension
NCOL(1:12) # 1
NROW(1:12) # 12, the length() of the vector
```

```
## as.matrix() produces 1-column matrices from 0-length vectors,
## and so does cbind() :
dim(as.matrix(numeric())) # 0 1
dim(  cbind(numeric())) # ditto
NCOL(numeric()) # 1
## However, as.matrix(NULL) fails and cbind(NULL) gives NULL, hence for
## consistency:
NCOL(NULL)      # 0
## (This gave 1 in R < 4.4.0.)
```

ns=dblcolon

pkg::name
pkg:::name

pkg
name

pkg::namenamenamepkgpkg:::namenamename

pkg::namepackage:pkg

:::maintainer

[getloadNamespaceasNamespace](#)

base::log
base::"+"

Beware -- use ':::' at your own risk! (see "Details")
stats::coef.default

ns-hooks

.onLoad(libname, pkgname)
.onAttach(libname, pkgname)
.onUnload(libpath)
.onDetach(libpath)
.Last.lib(libpath)

libname
pkgname
libpath

`loadNamespace.onLoad`
`libraryattachNamespace.onAttach`
`.onDetach.Last.libdetach.onAttachtryCatch`
`unloadNamespace.onUnload`
`.onLoad.onUnload`
`.onLoad.onUnload.onAttach.onDetach.Last.lib`
`.Last`
`.onLoad.onUnloaduseDynLibNAMESPACE.onLoad.onUnload.onAttach`

`.onAttach.onLoadpackageStartupMessage`
`libraryrequireDependsDESCRIPTION`
`libraryhelpformatpackageStartupMessage`
`installed.packages`
`library.dynam.onLoaduseDynLibNAMESPACE.onAttachlibrary.dynam.unload.Last.lib`
`.onDetach.onUnload`

`setHook`
`reg.finalizer`
`loadNamespace`

`ns-load`

```
attachNamespace(ns, pos = 2L, depends = NULL, exclude, include.only)
loadNamespace(package, lib.loc = NULL,
               keep.source = getOption("keep.source.pkgs"),
               partial = FALSE, versionCheck = NULL,
               keep.parse.data = getOption("keep.parse.data.pkgs"))
requireNamespace(package, ..., quietly = FALSE)
loadedNamespaces()
unloadNamespace(ns)
isNamespaceLoaded(name)
```

```

ns
pos
depends          NULL.Depends
package
lib.loc
keep.source
keep.parse.data

partial
versionCheck    NULLopversion
quietly
name            as.symbol"stats"
excludeinclude.only
               library
...            loadNamespace

loadNamespaceattachNamespacelibrary
loadNamespacelibraryloadNamespace.onLoad
library
loadNamespaceattachNamespacelibrary.onAttach
requireNamespaceloadNamespacerequire
loadedNamespaces
isNamespaceLoaded(pkg)pkg %in% loadedNamespaces()
unloadNamespacedetach.onDetach.Last.lib.onUnload
detach

attachNamespace
loadNamespace
requireNamespaceTRUEFALSE
loadedNamespacescharacter
unloadNamespaceNULL

loadNamespaceloadNamespace::load_R_TRACE_LOADNAMESPACE_124

```

[getNamespaceasNamespacetopenv.onLoadenvironment](#)

```
(lns <- loadedNamespaces())
statL <- isNamespaceLoaded("stats")
stopifnot( identical(statL, "stats" %in% lns) )

## The string "foo" and the symbol 'foo' can be used interchangeably here:
stopifnot( identical(isNamespaceLoaded( "foo" ), FALSE),
            identical(isNamespaceLoaded(quote(foo)), FALSE),
            identical(isNamespaceLoaded(quote(stats)), statL))

hasS <- isNamespaceLoaded("splines") # (to restore if needed)
Sns <- asNamespace("splines") # loads it if not already
stopifnot( isNamespaceLoaded("splines"))
if (is.null(try(unloadNamespace(Sns)))) # try unloading the NS 'object'
stopifnot( ! isNamespaceLoaded("splines"))
if (hasS) loadNamespace("splines") # (restoring previous state)
```

ns-topenv

[environment](#)envir

```
topenv(envir = parent.frame(),
        matchThisEnv = getOption("topLevelEnvironment"))
```

envir

```
matchThisEnv    topLevelEnvironmentsys.sourceNULLemptyenv()
```

topenv[environment](#)envir.[GlobalEnv](#)[search.GlobalEnv](#)

[environment](#)parent.env()[loadNamespace](#)

```
topenv(.GlobalEnv)
topenv(new.env()) # also global env
topenv(environment(ls))# namespace:base
topenv(environment(lm))# namespace:stats
```

NULL

NULLNULL

NULL
as.null(x, ...)
is.null(x)

x
...

NULLNULL[[]]NULLNULL
NULLNULL
NULLattrattributesstructure

as.nullNULL
is.nullTRUENULLFALSE

is.null

`%%L %||% R if(!is.null(L)) L else R`

```
is.null(list())      # FALSE (on purpose!)
is.null(pairlist()) # TRUE
is.null(integer(0)) # FALSE
is.null(logical(0)) # FALSE
as.null(list(a = 1, b = "c"))
```

numeric

"numeric" is numeric

```
numeric(length = 0)
as.numeric(x, ...)
is.numeric(x)
```

length

x

...

numericdouble0

as.numericas.doubleas.double

is.numericprimitiveis.double"complex""Date""POSIXt""difftime" is.numericdouble
integer

numericas.numericdouble

```
is.numericTRUE"numeric""double""integer"FALSEis.integer(x) || is.double(x)
(mode(x) == "numeric") && !is.factor(x)
```

xfactoras.numericfactorlevelsfactor

as.numericis.numericsetMethod

as.numericas.doubleas.numeric

doublenumericreal

doublenumeric"numeric"

"numeric" is.numericas.numericas.double

doubleintegerstorage.mode


```
## Conversion does trim whitespace; non-numeric strings give NA + warning
as.numeric(c("-.1", " 2.7 ", "B"))

## Numeric values are sometimes accidentally converted to factors.
## Converting them back to numeric is trickier than you'd expect.
f <- factor(5:10)
as.numeric(f) # not what you might expect, probably not what you want
## what you typically meant and want:
as.numeric(as.character(f))
## the same, considerably more efficient (for long vectors):
as.numeric(levels(f))[f]
```

NumericConstants

```
InfNaNtypeof(.)      "double" scanas.doubleinfinityInfNA_real_NA_integer_ "double"
"integer" L
0x0X0-9 a-f A-F .Pp0x123p456291  $\times 2^{456}$ 
Ee
Inf0.0
i
Linteger". "
```

```
.Machine$double.xminscan
```

```
SyntaxcomplexQuotesReserved
```

```
## You can create numbers using fixed or scientific formatting.
2.1
2.1e10
-2.1E-10

## The resulting objects have class numeric and type double.
class(2.1)
typeof(2.1)

## This holds even if what you typed looked like an integer.
```

```

class(2)
typeof(2)

## If you actually wanted integers, use an "L" suffix.
class(2L)
typeof(2L)

## These are equal but not identical
2 == 2L
identical(2, 2L)

## You can write numbers between 0 and 1 without a leading "0"
## (but typically this makes code harder to read)
.1234

sqrt(1i) # remember elementary math?
utils::str(0xA0)
identical(1L, as.integer(1))

## You can combine the "0x" prefix with the "L" suffix :
identical(0xFL, as.integer(15))

```

numeric_version

```

numeric_version(x, strict = TRUE)
package_version(x, strict = TRUE)
R_system_version(x, strict = TRUE)
getRversion()
as.numeric_version(x)
as.package_version(x)
is.numeric_version(x)
is.package_version(x)

```

```

x                package_versionR.versionas.numeric_versionas.package_version
                is.numeric_versionis.package_version

strict

```

DESCRIPTION.-

```

numeric_versionpackage_versionR_system_version
getRversion
[[

```

[compareVersionpackageVersionR.versiongetRversion\(\)](#)

```

x <- package_version(c("1.2-4", "1.2-3", "2.1"))
x < "1.4-2.3"
c(min(x), max(x))
x[2, 2]
x$major
x$minor

if(getRversion() <= "2.5.0") { ## work around missing feature
  cat("Your version of R, ", as.character(getRversion()),
      ", is outdated.\n",
      "Now trying to work around that ...\n", sep = "")
}

x[[1]]
x[[c(1, 3)]] # '4' as a numeric version
x[1, 3]      # same
x[[1, 3]]    # 4 as an integer

x[[2, 3]] <- 0 # zero the patchlevel
x[[c(2, 3)]] <- 0 # same
x

x[[3]] <- "2.2.3"
x

x <- c(x, package_version("0.0"))
is.na(x)[4] <- TRUE
stopifnot(identical(is.na(x), c(rep(FALSE,3), TRUE)),
           anyNA(x))

```

octmode

```

as.octmode(x)

## S3 method for class 'octmode'
as.character(x, keepStr = FALSE, ...)

## S3 method for class 'octmode'
format(x, width = NULL, ...)

## S3 method for class 'octmode'
print(x, ...)

x          "octmode"
keepStr    logical TRUE
width      NULL
...

```

```
"octmode"755[
as.character(x)attributeskeepStr=TRUEdimdimnamesnamesformat()width = NULL
as.octmode"integer""double"0-7NA"octmode"
!|&
```

```
file.info
```

```
hexmodesprintfstrtoi
```

```
(on <- as.octmode(c(16, 32, 127:129))) # "020" "040" "177" "200" "201"
unclass(on[3:4]) # subsetting
```

```
## manipulate file modes
fmode <- as.octmode("170")
(fmode | "644") & "755"
```

```
(umask <- Sys.umask()) # depends on platform
c(fmode, "666", "755") & !umask
```

```
om <- as.octmode(1:12)
om # print()s via format()
stopifnot(nchar(format(om)) == 2)
om[1:7] # *no* leading zeroes!
stopifnot(format(om[1:7]) == as.character(1:7))
om2 <- as.octmode(c(1:10, 60:70))
om2 # prints via format() -> with 3 octals
stopifnot(nchar(format(om2)) == 3)
as.character(om2) # strings of length 1, 2, 3
```

```
## Integer arithmetic (remaining "octmode"):
om^2
om * 64
-om
(fac <- factorial(om)) # !1, !2, !3, !4 .. in hexadecimal
as.integer(fac) # indeed the same as factorial(1:12)
```

```
on.exit
```

```
on.exit
```

```
on.exit()on.exit
```

```
on.exit(expr = NULL, add = FALSE, after = TRUE)
```

```
expr
add          exprafterexpr
after        addafterexpr
```

```
expron.exiton.exitexprsubstitute
on.exitadd = TRUE
addafter
```

```
NULL
```

```
sys.on.exiton.exit()sys.on.exit()
```

```
require(graphics)
```

```
opar <- par(mai = c(1,1,1,1))
on.exit(par(opar))
```

Ops.Date

```
"Date"
Ops+-Date
```

```
date + x
x + date
date - x
date1 lop date2
```

```
date          "Date"
date1date2    as.Date
x             "difftime"
lop           ==!<=>>=
```

```
xDates
```

```
(z <- Sys.Date())
z + 10
z < c("2009-06-01", "2010-01-01", "2015-01-01")
```

options

```
options(...)

getOption(x, default = NULL)

.Options

...                name = value

x
default

options()getOption("width")options("width")

getOptionxdefaultNULL
options()options(name)NULL

add.smooth TRUEplot.lm
askYesNo askYesNo
browserNLdisabled "n"
catch.script.errors interactive()R CMD BATCH <script>.Rtraceback()
checkPackageLicense loadNamespace
check.bounds FALSElistx <- 1:3; x[5] <- 6
CBoundsCheck .C.Fortran
      R_C_BOUNDS_CHECKyes
conflicts.policy libraryrequirelibrary
continue
defaultPackages R_DEFAULT_PACKAGESc("datasets",      "utils",      "grDevices",
      "graphics", "stats", "methods")R_DEFAULT_PACKAGESNULL.Rprofile
deparse.cutoff deparse60
deparse.max.lines browsertraceback.max.linestraceback()
traceback.max.lines traceback
digits signifprint.default
digits.secs NULLstrftime
```

```

download.file.extra download.file
download.file.method download.file"internal""wininet""libcurl""wget""curl"
  method = "auto"download.file
echo --no-echoFALSETRUE
encoding "native.enc"connections
error stopstopdump.framesrecoveroptions(error = utils::recover).Rprofile
expressions --max-ppsizeMemoryCstack_info
interrupt
keep.parse.data keep.sourcegetParseData()keep.sourceTRUE
keep.parse.data.pkgs keep.parse.dataFALSER_KEEP_PKG_PARSE_DATAyes
keep.source TRUEdeparse(fn, control = "useSource")
  interactive()TRUE
keep.source.pkgs keep.sourceFALSER_KEEP_PKG_SOURCEyes
matprod %%%crossprodtcrossprod
  "internal" NaNInfsumcolSumslong doublecapabilities
  "default" NaNInfNaNInf"default"NaNInfdouble
  "blas" NaNInfNaNInf
  "default.simd" "default"NaNInf"default"
max.print 99999printshowmax.print
OutDec formatformatCas.charactersprintf
pager file.show
  /bin/pagerPAGERless
  "internal""console"
  (files, header,title, delete.file)file.show
papersize postscriptR_PAPERSIZE
  LC_PAPER
  "a4""letter"
PCRE_limit_recursion grep(perl = TRUE)
  pcre_confighttps://www.pcre.org/original/doc/html/pcrestack.htmlNA
PCRE_study grep(perl = TRUE)10
PCRE_use_JIT grep(perl = TRUE)strsplit(perl = TRUE)
pdfviewer R_PDFVIEWER

  open.exe
printcmd postscriptR_PRINTCMDstdin"lpr"
prompt " "
rl_word_breaks " \t\n\"\\'`>=<=,;|&{()}"

  " \t\n\"\\'`>=<=+-*%,;|&{()}"
save.defaultssave.image.defaults save
scipen scipen0-6:319warning
setWidthOnResize TRUEreadlinewidth

```

```

showWarnCallsshowErrorCalls warningstopconditionCallNULL
showNCalls
show.error.locations TRUE"top"
show.error.messages try
texi2dvi texi2dvitexi2pdf
      R_TEXI2DVICMDTEXI2DVIitexi2dvi"emulation"
timeout R_DEFAULT_INTERNET_TIMEOUTdownload.fileconnections
topLevelEnvironment topenvsys.source
url.method url"default""internal"
useFancyQuotes sQuotedQuoteRd2txtTRUEFALSE"TeX""UTF-8"
verbose TRUE--verbose
warn warnwarnlast.warningwarningswarnwarn
warnPartialMatchArgs
warnPartialMatchAttr attr
warnPartialMatchDollar $
warning.expression warn
warning.length
nwarnings warn = 0
width cat

```

Print.h

add.smooth	TRUE
check.bounds	FALSE
continue	"+" "
digits	7
echo	TRUE
encoding	"native.enc"
error	NULL
expressions	5000
keep.source	interactive()
keep.source.pkgs	FALSE
max.print	99999
OutDec	". "
prompt	"> "
scipen	0
show.error.messages	TRUE
timeout	60
verbose	FALSE
warn	0
warning.length	1000
width	80


```
bitmapType png"cairo""quartz"
device X11windowsquartzpdf
      R_INTERACTIVE_DEVICER_DEFAULT_DEVICE
      windowsquartzX11DISPLAY
device.ask.default devAskNewPage("ask")
locatorBell locatoridentifyTRUEX11windows
windowsTimeouts windows

max.contour.segments 25000contourcontourLines
```

```
contrasts contrastsaovlmc("unordered", "ordered")
na.action NAna.actionna.pass
show.coef.Pvalues printCoefmat
show.nls.convergence nlsTRUE
show.signif.stars printCoefmat
str.dendrogram.last str.dendrogram
ts.eps ts1e-05
ts.S.compat logplot.spec
```

```
BioC_mirror setRepositories"https://bioconductor.org"
           "https://bioconductor.statistik.tu-dortmund.de"chooseBioCmirror
browser browseURLbrowseURL
ccaddress create.postbug.reporthehelp.requestFALSE""
citation.bibtex.max bibentrycitation
de.cellwidth dataentryNA
demo.ask askdemo
editor editfile.editEDITORVISUALvi
example.ask askexample
help.ports startDynamicHelp
help.search.types help.search??
help.try.all.packages help
help_type help?
help.htmlmath texmathRd2HTML"katex"NULL"mathjax"
help.htmltoc tocRd2HTML
```

```

HTTPUserAgent download.fileurlcurlGetHeadersNULL"R ( )"libcurl"libcurl/"libcurl
install.lock install.packagesR CMD INSTALL"pkglock"
internet.info "internal""libcurl""internal"
install.packages.check.source install.packagesupdate.packages"yes""no""yes"
install.packages.compile.from.source install.packages(type
  "both")update.packages"never""interactive""never""always"
  R_COMPILE_AND_INSTALL_PACKAGES"interactive"install.packages"never"make
  MAKE
mailer create.postbug.reporthehelp.request
menu.graphics TRUEselect.listchooseCRANmirrorsetRepositoriesshelp
Ncpus  $n \geq 1$ install.packagesNcpus = getOption("Ncpus", 1L)
pkgType install.packages
  "win.binary""source""both"
  "source""mac.binary""mac.binary.big-sur-arm64""both"
  "binary""both"install.packages
repos available.packagesrepositoriesR_REPOSITORIESNULLR.home("etc")
  c(CRAN="@CRAN@")
  local({
    r <- getOption("repos")
    r["CRAN"] <- "https://my.local.cran"
    options(repos = r)
  })
  .Rprofilerepositories
  setRepositories
str strstrOptions()
SweaveHooksSweaveSyntax Sweave
unzip unzipunzip"internal"
  R_UNZIPCMDetc/Renvironunzip""
  "internal"

mc.cores MC_CORES2

dvipscmd "dvips"

warn.FPU

.Optionsoptions()options()
NULL

```

```

op <- options(); utils::str(op) # op is a named list

getOption("width") == options()$width # the latter needs more memory
options(digits = 15)
pi

# set the editor, and save previous value
old.o <- options(editor = "nedit")
old.o

options(check.bounds = TRUE, warn = 1)
x <- NULL; x[4] <- "yes" # gives a warning

options(digits = 5)
print(1e5)
options(scipen = 3); print(1e5)

options(op) # reset (all) initial options
options("digits")

## Not run: ## set contrast handling to be like S
options(contrasts = c("contr.helmert", "contr.poly"))

## End(Not run)

## Not run: ## on error, terminate the R session with error status 66
options(error = quote(q("no", status = 66, runLast = FALSE)))
stop("test it")

## End(Not run)

## Not run: ## Set error actions for debugging:
## enter browser on error, see ?recover:
options(error = recover)
## allows to call debugger() afterwards, see ?debugger:
options(error = dump.frames)
## A possible setting for non-interactive sessions
options(error = quote({dump.frames(to.file = TRUE); q()}))

## End(Not run)

# Compare the two ways to get an option and use it
# accounting for the possibility it might not be set.
if(as.logical(getOption("performCleanup", TRUE)))
  cat("do cleanup\n")

## Not run:
# a clumsier way of expressing the above w/o the default.
tmp <- getOption("performCleanup")
if(is.null(tmp))
  tmp <- TRUE
if(tmp)

```

```

cat("do cleanup\n")

## End(Not run)

```

order

```
ordersort.list
```

```

order(..., na.last = TRUE, decreasing = FALSE,
      method = c("auto", "shell", "radix"))

sort.list(x, partial = NULL, na.last = TRUE, decreasing = FALSE,
         method = c("auto", "shell", "quick", "radix"))

...
x          method"shell""quick"x"auto""radix"order(x,..)
partial    NULL
decreasing  "radix"...
na.last     NATRUEFALSENA
method      "auto""radix"231"shell""shell""quick""radix"sort

method = "quick"

"radix"Comparison
"shell""radix""radix"sort"quick"sort.listxna.last = NA"radix"
partial = NULL
xtfrmis.numeric(x)

231

...decreasingdo.call('order', df_obj)df_objdf_objunname

sort.listsortx
na.last = NAsort.listNAorderNA

x[order(x, na.last = NA)]
zz <- x[!is.na(x)]; zz[sort.list(x, na.last = NA)]

NAx
method = "radix"

```

sortrankxtfrm

```
require(stats)

(ii <- order(x <- c(1,1,3:1,1:4,3), y <- c(9,9:1), z <- c(2,1:9)))
## 6 5 2 1 7 4 10 8 3 9
rbind(x, y, z)[,ii] # shows the reordering (ties via 2nd & 3rd arg)

## Suppose we wanted descending order on y.
## A simple solution for numeric 'y' is
rbind(x, y, z)[, order(x, -y, z)]
## More generally we can make use of xtfrm
cy <- as.character(y)
rbind(x, y, z)[, order(x, -xtfrm(cy), z)]
## The radix sort supports multiple 'decreasing' values:
rbind(x, y, z)[, order(x, cy, z, decreasing = c(FALSE, TRUE, FALSE),
                      method="radix")]

## Sorting data frames:
dd <- transform(data.frame(x, y, z),
                 z = factor(z, labels = LETTERS[9:1]))
## Either as above {for factor 'z' : using internal coding}:
dd[ order(x, -y, z), ]
## or along 1st column, ties along 2nd, ... *arbitrary* no.{columns}:
dd[ do.call(order, dd), ]

set.seed(1) # reproducible example:
d4 <- data.frame(x = round( rnorm(100)), y = round(10*runif(100)),
                 z = round( 8*rnorm(100)), u = round(50*runif(100)))
(d4s <- d4[ do.call(order, d4), ])
(i <- which(diff(d4s[, 3]) == 0))
# in 2 places, needed 3 cols to break ties:
d4s[ rbind(i, i+1), ]

## rearrange matched vectors so that the first is in ascending order
x <- c(5:1, 6:8, 12:9)
y <- (x - 5)^2
o <- order(x)
rbind(x[o], y[o])

## tests of na.last
a <- c(4, 3, 2, NA, 1)
b <- c(4, NA, 2, 7, 1)
z <- cbind(a, b)
(o <- order(a, b)); z[o, ]
(o <- order(a, b, na.last = FALSE)); z[o, ]
(o <- order(a, b, na.last = NA)); z[o, ]
```

```
## speed examples on an average laptop for long vectors:
## factor/small-valued integers:
x <- factor(sample(letters, 1e7, replace = TRUE))
system.time(o <- sort.list(x, method = "quick", na.last = NA)) # 0.1 sec
stopifnot(!is.unsorted(x[o]))
system.time(o <- sort.list(x, method = "radix")) # 0.05 sec, 2X faster
stopifnot(!is.unsorted(x[o]))
## large-valued integers:
xx <- sample(1:200000, 1e7, replace = TRUE)
system.time(o <- sort.list(xx, method = "quick", na.last = NA)) # 0.3 sec
system.time(o <- sort.list(xx, method = "radix")) # 0.2 sec
## character vectors:
xx <- sample(state.name, 1e6, replace = TRUE)
system.time(o <- sort.list(xx, method = "shell")) # 2 sec
system.time(o <- sort.list(xx, method = "radix")) # 0.007 sec, 300X faster
## double vectors:
xx <- rnorm(1e6)
system.time(o <- sort.list(xx, method = "shell")) # 0.4 sec
system.time(o <- sort.list(xx, method = "quick", na.last = NA)) # 0.1 sec
system.time(o <- sort.list(xx, method = "radix")) # 0.05 sec, 2X faster
```

outer

```
XYAc(dim(X), dim(Y))A[c(arrayindex.x, arrayindex.y)] = FUN(X[arrayindex.x],
Y[arrayindex.y], ...)
```

```
outer(X, Y, FUN = "*", ...)
X %o% Y
```

XY	FUN
FUN	<code>match.fun</code> "*"
...	FUN

```
XYFUNrepXYFUN
FUN...
XYXY
FUN = "*"as.vector(X) %*% t(as.vector(Y))
%o%outer(x, y, "*")
```

[%*%kroneckerouterVectorize](#)

```
x <- 1:9; names(x) <- x
# Multiplication & Power Tables
x %o% x
y <- 2:8; names(y) <- paste(y, ":", sep = "")
outer(y, x, `^`)

outer(month.abb, 1999:2003, FUN = paste)

## three way multiplication table:
x %o% x %o% y[1:3]
```

Paren

```
({.Primitive
(function(x) x{

( ... )

{ ... }

(
{
```

[ifreturn](#)

[Syntax](#)

```
f <- get("")
e <- expression(3 + 2 * 4)
identical(f(e), e)

do <- get("{}")
do(x <- 3, y <- 2*x-3, 6-x-y); x; y

## note the differences
(2+3)
{2+3; 4+5}
(invisible(2+3))
{invisible(2+3)}
```

parse

parse()[expressioncall](#)

str2expression(s)str2lang(s)parse(text=s, keep.source=FALSE)s

```
parse(file = "", n = NULL, text = NULL, prompt = "?",
      keep.source = getOption("keep.source"), srcfile,
      encoding = "unknown")
```

str2lang(s)

str2expression(text)

file file""textNULL

n nNULLNA

text

prompt NULLgetOption("prompt")

keep.source TRUE

srcfile NULL[srcfile](#)

encoding "latin1""UTF-8"con[options](#)(encoding=)[file](#)encoding = "latin1"
encoding = "UTF-8"

s [character](#)1

parse(...) textfile

```
  n = NULLn = 1n < 0
```

```
  srcfilekeep.sourceTRUEsrcfile"<text>"filekeep.sourceTRUEtextsrcfile
```

```
  srcfilecopyfilessrcfile
```

```
  srcfilesrcfilessrcfile
```

```
str2expression(s) charactersstr2expression(s)parse(text = s, keep.source=FALSE)
  typeofclassexpression
```

```
str2lang(s) charactersstr2lang(s)parse(text = s, keep.source=FALSE)[[1]]s
  parse(*)callsymbolnameNULL21LTRUEstr2lang(.)
```

```
str2lang()str2expression()
```

```
parse()str2expression()"expression"parse()n
```

```
str2lang(s)scall
```

```
srcfileNULL"srcref"srcref"srcfile"srcfile"wholeSrcref"srcref"srcfile"
getParseData
```

```
encoding"latin1""UTF-8"text
```



```
parseSrcfilesrcfiletext
getParseDatasrcfile"parent"
```

```
parse(text = *, ..)str2lang()str2expression()call(..)as.call()
```

```
scansourceevaldeparse
setBreakpointRprofgetSrcrefgetParseData
```

```
fil <- tempfile(fileext = ".Rdmped")
cat("x <- c(1, 4)\n x ^ 3 -10 ; outer(1:7, 5:9)\n", file = fil)
# parse 3 statements from our temp file
parse(file = fil, n = 3)
unlink(fil)

## str2lang(<string>) || str2expression(<character>) :
stopifnot(exprs = {
  identical( str2lang("x[3] <- 1+4"), quote(x[3] <- 1+4))
  identical( str2lang("log(y)"),      quote(log(y)) )
  identical( str2lang("abc" ),        quote(abc) -> qa)
  is.symbol(qa) & !is.call(qa)         # a symbol/name, not a call
  identical( str2lang("1.375" ), 1.375) # just a number, not a call
  identical( str2expression(c("# a comment", "", "42")), expression(42) )
})

# A partial parse with a syntax error
txt <- "
x <- 1
an error
"
sf <- srcfile("txt")
tryCatch(parse(text = txt, srcfile = sf), error = function(e) "Syntax error.")
getParseData(sf)
```

paste

collapse

```
paste(..., sep = " ", collapse = NULL, recycle0 = FALSE)
paste0(...,          collapse = NULL, recycle0 = FALSE)
```

```
...
sep          NA_character_
collapse     NA_character_collapsecharacter
recycle0     logicalcharacter(0)collapse0recycle0""
```

```
pasteas.charactersep
""recycle0TRUEcollapseNULL
paste()NA_character_"NA"paste("the value of p is ", p)
paste0(..., collapse)paste(..., sep = "", collapse)
collapsecollapse
```

```
collapse""
"bytes"Encoding
"bytes""bytes"collapse
```

```
toStringpaste(*, collapse=", ")as.charactersubstrncharstrsplitcatsprintf
paste
```

```
## When passing a single vector, paste0 and paste work like as.character.
paste0(1:12)
paste(1:12)      # same
as.character(1:12) # same
```

```
## If you pass several vectors to paste0, they are concatenated in a
## vectorized way.
(nth <- paste0(1:12, c("st", "nd", "rd", rep("th", 9))))
```

```
## paste works the same, but separates each input with a space.
## Notice that the recycling rules make every input as long as the longest input.
paste(month.abb, "is the", nth, "month of the year.")
paste(month.abb, letters)
```

```
## You can change the separator by passing a sep argument
## which can be multiple characters.
paste(month.abb, "is the", nth, "month of the year.", sep = "_*_")
```

```
## To collapse the output into a single string, pass a collapse argument.
paste0(nth, collapse = ", ")
```

```
## For inputs of length 1, use the sep argument rather than collapse
paste("1st", "2nd", "3rd", collapse = ", ") # probably not what you wanted
```

```

paste("1st", "2nd", "3rd", sep = ", ")

## You can combine the sep and collapse arguments together.
paste(month.abb, nth, sep = ": ", collapse = "; ")

## Using paste() in combination with strwrap() can be useful
## for dealing with long strings.
(title <- paste(strwrap(
  "Stopping distance of cars (ft) vs. speed (mph) from Ezekiel (1930)",
  width = 30), collapse = "\n"))
plot(dist ~ speed, cars, main = title)

## zero length arguments recycled as `""` -- NB: `{}` <==> character(0) here
paste({}, 1:2)

## 'recycle0 = TRUE' allows standard vectorized behaviour, i.e., zero-length
## recycling resulting in zero-length result character(0):
valid <- FALSE
val <- pi
paste("The value is", val[valid], "-- not so good!") # -> ".. value is -- not .."
paste("The value is", val[valid], "-- good: empty!", recycle0=TRUE) # -> character(0)

## When 'collapse = <string>', result is (length 1) string in all cases
paste("foo", {}, "bar", collapse = "|") # |--> "foo bar"
paste("foo", {}, collapse = "|", recycle0 = TRUE) # |--> ""
## If all arguments are empty (and collapse a string), "" results always
paste(collapse = "|")
paste(collapse = "|", recycle0 = TRUE)
paste({}, collapse = "|")
paste({}, collapse = "|", recycle0 = TRUE)

```

path.expand

path.expand(path)

path

```

~useruser
HOME

```

```

rw-FAQR_USERHOMER_USERpath.expand

```

basenamenormalizePathfile.path

path.expand("~/foo")

pcre_config

pcre_config()

UTF-8

Unicode properties

\p{xx}\P{xx}

JIT

arm64

stack

TRUE<https://www.pcre.org/original/doc/html/pcrestack.html>FALSE

extSoftVersion

pcre_config()

pipeOp

lhs |> rhs

lhs

rhs

```

lhsrhs
lhsx |> f(y)f(x, y)
rhs+if
_rhslhsrhs
_.$coef_.$coef[[2]]$[[@

x |> f(y)f(x, y)rhs

```

```

# simple uses:
mtcars |> head()           # same as head(mtcars)
mtcars |> head(2)          # same as head(mtcars, 2)
mtcars |> subset(cyl == 4) |> nrow() # same as nrow(subset(mtcars, cyl == 4))

# to pass the lhs into an argument other than the first, either
# use the _ placeholder with a named argument:
mtcars |> subset(cyl == 4) |> lm(mpg ~ disp, data = _)
# or use an anonymous function:
mtcars |> subset(cyl == 4) |> (function(d) lm(mpg ~ disp, data = d))()
mtcars |> subset(cyl == 4) |> (\(d) lm(mpg ~ disp, data = d))()
# or explicitly name the argument(s) before the "one":
mtcars |> subset(cyl == 4) |> lm(formula = mpg ~ disp)

# using the placeholder as the head of an extraction chain:
mtcars |> subset(cyl == 4) |> lm(formula = mpg ~ disp) |> _.$coef[[2]]

# the pipe operator is implemented as a syntax transformation:
quote(mtcars |> subset(cyl == 4) |> nrow())

# regular R evaluation semantics apply
stop() |> (function(...) {})( ) # stop() is not used on RHS so is not evaluated

```

plot

```

plot.defaultplotfunctiondata.framedensitymethods(plot)
par

```

```
plot(x, y, ...)
```

```
x          plot
y          x
...        par
          type
            "p"
            "l"
            "b"
            "c""b"
            "o"
            "h"
            "s"
            "S"
            "n"
          type = "punkte" type = "p" plot.factor
main title
sub title
xlab title
ylab title
asp y/x plot.window
```

```
(x1,y1)(x2,y2)x1 < x2 type = "s" type = "S"
```

```
plot
```

```
plot.default plot.formula points lines pars smooth Scatter() plot()
contour persp image
```

```
require(stats) # for lowess, rpois, rnorm
require(graphics) # for plot methods
plot(cars)
lines(lowess(cars))
```

```
plot(sin, -pi, 2*pi) # see ?plot.function
```

```
## Discrete Distribution Plot:
plot(table(rpois(100, 5)), type = "h", col = "red", lwd = 10,
      main = "rpois(100, lambda = 5)")
```

```
## Simple quantiles/ECDF, see ecdf() {library(stats)} for a better one:
plot(x <- sort(rnorm(47)), type = "s", main = "plot(x, type = \"s\")")
points(x, cex = .5, col = "dark red")
```

`pmatch`

`pmatch`

```
pmatch(x, table, nomatch = NA_integer_, duplicates.ok = FALSE)
```

```
x                as.character
```

```
table
```

```
nomatch          integer
```

```
duplicates.ok    table
```

```
duplicates.okxtablex
```

```
duplicates.okFALSEtable
```

```
charmatchpmatchduplicates.ok
```

```
NA"NA"
```

```
NAnomatch = NAxtablenomatch
```

```
matchcharmatchmatch.argmatch.funmatch.callstartsWithgrep
```

```
pmatch("", "") # returns NA
```

```
pmatch("m", c("mean", "median", "mode")) # returns NA
```

```
pmatch("med", c("mean", "median", "mode")) # returns 2
```

```
pmatch(c("", "ab", "ab"), c("abc", "ab"), duplicates.ok = FALSE)
```

```
pmatch(c("", "ab", "ab"), c("abc", "ab"), duplicates.ok = TRUE)
```

```
## compare
```

```
charmatch(c("", "ab", "ab"), c("abc", "ab"))
```

polyroot

polyroot(z)

z

$n - 1$

$$p(x) = z_1 + z_2x + \cdots + z_nx^{n-1}$$

z[1:n]polyrootn - 1p(x)

z

$n - 1nz$

unirootcomplexzero

```
polyroot(c(1, 2, 1))
round(polyroot(choose(8, 0:8)), 11) # guess what!
for (n1 in 1:4) print(polyroot(1:n1), digits = 4)
polyroot(c(1, 2, 1, 0, 0)) # same as the first
```

```
pos.to.env
```

```
pos.to.env(x)
```

```
x          1length(search())-1
```

```
getlspos.to.env
-1
```

```
pos.to.env(1) # R_GlobalEnv
# the next returns the base environment
pos.to.env(length(search()))
```

```
pretty
```

```
n+1x
```

```
pretty(x, ...)
```

```
## Default S3 method:
pretty(x, n = 5, min.n = n %% 3, shrink.sml = 0.75,
       high.u.bias = 1.5, u5.bias = .5 + 1.5*high.u.bias,
       eps.correct = 0, f.min = 2^-20, ...)
```

```
.pretty(x, n = 5L, min.n = n %% 3, shrink.sml = 0.75,
        high.u.bias = 1.5, u5.bias = .5 + 1.5*high.u.bias,
        eps.correct = 0L, f.min = 2^-20, bounds = TRUE)
```

```
x          as.numeric
n
min.n      min.n == 0pretty(.)
shrink.sml range(x)
high.u.bias > 1bhigh.u.bias
u5.bias    u5.bias = .5 + 1.5*high.u.bias
eps.correct range(x)eps.correct >= 2eps.correct = 2
f.min      .Machine$double.xmincell $c_m$ unitcell $c_n$ warningf.min = 20
bounds     logicalrange(x)xbounds=FALSE.pretty()
...
```

```
prettyx
```

```
d <- max(x) - min(x) ≥ 0 dc <- d/nc <- max(abs(range(x)))*shrink.sml / min.nb  
10[log10(c)] b ≤ c < 10b
```

```
u{1,2,5,10}bc/b ∈ [1,10)h =high.u.biasf =u5.bias
```

```
GEPretty()R_pretty()
```

```
(min.n = 1, shrink.sml = 0.25, high.u.bias = 0.8, u5.bias = 1.7,  
f.min = 1.125, eps.correct = 2, bounds = FALSE)
```

```
pretty()npretty(..)R_pretty().pretty()listbounds=TRUElunbounds=FALSEnsnununitn*  
ninteger
```

```
axTicks
```

```
pretty(1:15)          # 0  2  4  6  8 10 12 14 16  
pretty(1:15, high.u.bias = 2) # 0  5 10 15  
pretty(1:15, n = 4)    # 0  5 10 15  
pretty(1:15 * 2)       # 0  5 10 15 20 25 30  
pretty(1:20)           # 0  5 10 15 20  
pretty(1:20, n = 2)    # 0 10 20  
pretty(1:20, n = 10)   # 0  2  4 ... 20
```

```
for(k in 5:11) {  
  cat("k=", k, ": "); print(diff(range(pretty(100 + c(0, pi*10^-k))))))}
```

```
##-- more bizarre, when min(x) == max(x):  
pretty(pi)
```

```
add.names <- function(v) { names(v) <- paste(v); v}  
utils::str(lapply(add.names(-10:20), pretty))  
## min.n = 0 returns a length-1 vector "if pretty":  
utils::str(lapply(add.names(0:20), pretty, min.n = 0))  
sapply( add.names(0:20), pretty, min.n = 4)
```

```
pretty(1.234e100)  
pretty(1001.1001)  
pretty(1001.1001, shrink.sml = 0.2)  
for(k in -7:3)  
  cat("shrink=", formatC(2^k, width = 9),":",  
      formatC(pretty(1001.1001, shrink.sml = 2^k), width = 6),"\n")
```

Primitive

`.Primitive`

`.Primitive(name)`

`name`

`.Primitive.Internalswitch`

`attributeswarning`

``name`get(name, envir = baseenv())`

`is.primitivetypeof.Internal`

```
mysqrt <- .Primitive("sqrt")
## be careful, this changes base R's sqrt() {until restarting R} !
try(structure(mysqrt, comment = "primitive function")) # deprecation warning; soon an error

.Internal # this one must be primitive!
`if` # need backticks
```

print

`printinvisible(x)class`

`print(x, ...)`

```
## S3 method for class 'factor'
print(x, quote = FALSE, max.levels = NULL,
      width = getOption("width"), ...)
```

```
## S3 method for class 'table'
print(x, digits = getOption("digits"), quote = FALSE,
      na.print = "", zero.print = "0",
      right = is.numeric(x) || is.complex(x),
      justify = "none", ...)
```

```
## S3 method for class 'function'
print(x, useSource = TRUE, ...)
```

```

x
...
quote
max.levels      0NULLmax.levelswidth
width           max.levels
digits          print.default
na.print        NULLNAprint.default
zero.print      0"."
right
justify         format
useSource       options(keep.source = TRUE)

```

```

print.defaultmethods("print")print
print.factorordered
print.tabletable
noquoteprint

```

```

print.defaultoptionsnoquote
catformatwrite.print.via.format

```

```

require(stats)

```

```

ts(1:20) #-- print is the "Default function" --> print.ts(.) is called
for(i in 1:3) print(1:i)

```

```

## Printing of factors
attenu$station ## 117 levels -> 'max.levels' depending on width

```

```

## ordered factors: levels  "l1 < l2 < .."
esoph$agegp[1:12]
esoph$alcgp[1:12]

```

```

## Printing of sparse (contingency) tables
set.seed(521)
t1 <- round(abs(rt(200, df = 1.8)))
t2 <- round(abs(rt(200, df = 1.4)))
table(t1, t2) # simple
print(table(t1, t2), zero.print = ".") # nicer to read

```

```

## same for non-integer "table":
T <- table(t2,t1)

```

```

T <- T * (1+round(rlnorm(length(T)))/4)
print(T, zero.print = ".") # quite nicer,
print.table(T[,2:8] * 1e9, digits=3, zero.print = ".")
## still slightly inferior to Matrix::Matrix(T) for larger T

## Corner cases with empty extents:
table(1, NA) # < table of extent 1 x 0 >

```

```
print.data.frame
```

```

## S3 method for class 'data.frame'
print(x, ..., digits = NULL,
      quote = FALSE, right = TRUE, row.names = TRUE, max = NULL)

```

```

x          data.frame
...        print
digits     print.default
quote
right
row.names
max        NULLNULLgetOption("max.print")

```

```

formatprint
quote = TRUE

```

```
data.frame
```

```

(dd <- data.frame(x = 1:8, f = gl(2,4), ch = letters[1:8]))
# print() with defaults
print(dd, quote = TRUE, row.names = FALSE)
# suppresses row.names and quotes all entries

```

print.default

print.defaultprint

```
## Default S3 method:
print(x, digits = NULL, quote = TRUE,
      na.print = NULL, print.gap = NULL, right = FALSE,
      max = NULL, width = NULL, useSource = TRUE, ...)
```

```
x
digits      digitsNULLgetOption("digits")signif
quote       character
na.print    NANULL
print.gap   ≤ 1024NULL
right
max         maxNULLgetOption("max.print")
width      NULLgetOption("width")
useSource
...
```

```
NANANAquote = FALSE<NA>na.printNULLNAquote
digits
scientificxformat(., scientific=FALSE)prI()scipenoptions(scipen = 12)
digitsprint.default
widthdeparse.cutoff
printshow
```

```
digitsdigits >= 16sprintf()
```

```
\a\b\f\n\r\t\v\\0\243
```

```
0x000x7f
\uxxxx\Uxxxxxxxx
\xab
```

```
printoptions"noquote"  
encodeString
```

```
pi  
print(pi, digits = 16)  
LETTERS[1:16]  
print(LETTERS, quote = FALSE)  
  
M <- cbind(I = 1, matrix(1:10000, ncol = 10,  
                        dimnames = list(NULL, LETTERS[1:10])))  
utils::head(M)          # makes more sense than  
print(M, max = 1000)    # prints 90 rows and a message about omitting 910  
  
(x <- 2^seq(-8, 30, by=1/4)) # auto-prints; by default all in "exponential" format  
prI <- function(x) noquote(format(x, scientific = FALSE))  
prI(x) # prints more "nicely" (using a bit more space)
```

prmatrix

```
prmatrix(x, rowlab =, collab =,  
         quote = TRUE, right = FALSE, na.print = NULL, ...)
```

```
x  
rowlabcollab  dimnames(x)  
quote         TRUE"character"  
right         TRUE"character"  
na.print      NANA  
...           print
```

```
prmatrixprint.matrix
```

```
x
```

```
print.defaultprint
```

```
prmatrix(m6 <- diag(6), rowlab = rep("", 6), collab = rep("", 6))

chm <- matrix(scan(system.file("help", "AnIndex", package = "splines"),
                           what = ""), , 2, byrow = TRUE)
chm # uses print.matrix()
prmatrix(chm, collab = paste("Column", 1:3), right = TRUE, quote = FALSE)
```

proc.time

proc.time

proc.time()

proc.timeprint

"proc_time"printsummary

NA

system.timegc.time

setTimeLimit

```
## a way to time an R expression: system.time is preferred
ptm <- proc.time()
for (i in 1:50) mad(stats::runif(500))
proc.time() - ptm
```

prod

prod

prod(..., na.rm = FALSE)

...

na.rm TRUEFALSE

na.rmFALSENANANA

[Summary](#)...

NULLnumeric(0)

"double"

[Summary](#)x, ..., na.rm

[sumcumprod](#)[cumsum](#)

prod

print(prod(1:7)) == print(gamma(8))

proportions

marginsx

proportions(x, margin = NULL)

prop.table(x, margin = NULL)

x [table](#)

margin 12c(1, 2)x[dimnames](#)

xmargin

prop.table

[marginSums](#)

[applysweep](#)

```
m <- matrix(1:4, 2)
m
proportions(m, 1)

DF <- as.data.frame(UCBAdmissions)
tbl <- xtabs(Freq ~ Gender + Admit, DF)
tbl
proportions(tbl, "Gender")
```

pushBack

```
pushBack(data, connection, newLine = TRUE,
          encoding = c("", "bytes", "UTF-8"))
pushBackLength(connection)
clearPushBack(connection)
```

data
connection
newLine
encoding

pushBack[readLines](#)[scan](#)

[scan](#)("")[stdin](#)

[Encoding](#)encoding = ""<U+xxxx>encoding = "UTF-8"encoding = "bytes"

```
pushBackclearPushBack()
pushBackLength
```

[connectionsreadLines](#)

```
zz <- textConnection(LETTERS)
readLines(zz, 2)
pushBack(c("aa", "bb"), zz)
pushBackLength(zz)
readLines(zz, 1)
pushBackLength(zz)
readLines(zz, 1)
readLines(zz, 1)
close(zz)
```

qr

qr

```
qr(x, ...)
## Default S3 method:
qr(x, tol = 1e-07 , LAPACK = FALSE, ...)
```

```
qr.coef(qr, y)
qr.qy(qr, y)
qr.qty(qr, y)
qr.resid(qr, y)
qr.fitted(qr, y, k = qr$rank)
qr.solve(a, b, tol = 1e-7)
## S3 method for class 'qr'
solve(a, b, ...)
```

```
is.qr(x)
as.qr(x)
```

```
x
tol          xLAPACKx
qr           qr
yb
a            qr.solve
k
LAPACK       x
...
```

$$Ax = bAb$$

```
qr.coefqr.residqr.fittedyqrNAqr.qyqr.qtyQ %*% yt(Q) %*% yQQ
dimnamesnamesxy
solve.qrsolveqrqr.solveasolve.qra
is.qrTRUExlistinherits"qr"
"qr"
x231
qr.fittedqr.resid
```

```
qr          xRQ
qraux       ncol(x)min(dim(x))Q
rank        x
pivot
"useLAPACK"TRUE
```

*)dqrdc2

```
LAPACK = FALSEqr()dqrdc2tol
```

eigendet

```
qrDQRDCdqrdc2DGEQP3ZGEQP3qr.coefqr.qyqr.qty
```

<https://netlib.org/lapack/><https://netlib.org/linpack/>

https://netlib.org/lapack/lug/lapack_lug.html

```
qr.Qqr.Rqr.Xlm.fitlsfiteigensvd
```

```
detqr
```

```

hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
h9 <- hilbert(9); h9
qr(h9)$rank          #--> only 7
qrh9 <- qr(h9, tol = 1e-10)
qrh9$rank            #--> 9
##-- Solve linear equation system H %*% x = y :
y <- 1:9/10
x <- qr.solve(h9, y, tol = 1e-10) # or equivalently :
x <- qr.coef(qrh9, y) #-- is == but much better than
                        #-- solve(h9) %*% y
h9 %*% x              # = y

```

```

## overdetermined system
A <- matrix(runif(12), 4)
b <- 1:4
qr.solve(A, b) # or solve(qr(A), b)
solve(qr(A, LAPACK = TRUE), b)
# this is a least-squares solution, cf. lm(b ~ 0 + A)

```

```

## underdetermined system
A <- matrix(runif(12), 3)
b <- 1:3
qr.solve(A, b)
solve(qr(A, LAPACK = TRUE), b)
# solutions will have one zero, not necessarily the same one

```

QR.Auxiliaries

```

qr.X(qr, complete = FALSE, ncol =)
qr.Q(qr, complete = FALSE, Dvec =)
qr.R(qr, complete = FALSE)

```

qr	qrfsfit
complete	<i>QXR</i>
ncol	1:nrow(qr\$qr) <i>X</i> completeFALSEmin(ncol(X), nrow(X)) <i>X</i> completeTRUE <i>X</i> ncol(X)
Dvec	<i>Q</i> 1

```

qr.XXncol(X) <- nrow(X)completeTRUEncol(X)X
qr.Qnrow(X)qrcompleteTRUEnrow(X)completeFALSEncol(X)DvecDvec
qr.Q(qr, *)qr.qy(qr, y)yqr.X(qr, *)qr.qy(qr, R)dimnames
qr.Ra <- qr(x)x[, a$pivot]nrow(X)ncol(X)completeTRUEFALSE

```

qrqr.qy

```
p <- ncol(x <- LifeCycleSavings[, -1]) # not the 'sr'
qrstr <- qr(x) # dim(x) == c(n,p)
qrstr $ rank # = 4 = p
Q <- qr.Q(qrstr) # dim(Q) == dim(x)
R <- qr.R(qrstr) # dim(R) == ncol(x)
X <- qr.X(qrstr) # X == x
range(X - as.matrix(x)) # ~ < 6e-12
## X == Q %*% R if there has been no pivoting, as here:
all.equal(unname(X),
          unname(Q %*% R))

# example of pivoting
x <- cbind(int = 1,
           b1 = rep(1:0, each = 3), b2 = rep(0:1, each = 3),
           c1 = rep(c(1,0,0), 2), c2 = rep(c(0,1,0), 2), c3 = rep(c(0,0,1),2))
x # is singular, columns "b2" and "c3" are "extra"
a <- qr(x)
zapsmall(qr.R(a)) # columns are int b1 c1 c2 b2 c3
a$pivot
pivI <- sort.list(a$pivot) # the inverse permutation
all.equal(x, qr.Q(a) %*% qr.R(a)) # no, no
stopifnot(
  all.equal(x[, a$pivot], qr.Q(a) %*% qr.R(a)), # TRUE
  all.equal(x, qr.Q(a) %*% qr.R(a)[, pivI])) # TRUE too!
```

quit

quitq

```
quit(save = "default", status = 0, runLast = TRUE)
  q(save = "default", status = 0, runLast = TRUE)
```

```
save      "no""yes""ask""default"
status    0
runLast    .Last()
```

```
save"no""yes""ask""default"
.Last().LastrunLast.Last.Last.sys().Last()runLast
.Last().Last.sys()runLastsavehistoryreg.finalizer.Last()
q("no", 1, FALSE)
status0:255
```

`.Lastreg.finalizer`

`R.appsave`

`.First`

```
## Not run: ## Unix-flavour example
.Last <- function() {
  graphics.off() # close devices before printing
  cat("Now sending PDF graphics to the printer:\n")
  system("lpr Rplots.pdf")
  cat("bye bye...\n")
}
quit("yes")
## End(Not run)
```

Quotes

-

```
\n
\r
\t
\b
\a
\f
\v
\\      \
\'      '
\"      "
\'      '
\nnn
```

```
\xnn
\unnnn
\Unnnnnnnn
```

```
\u{nnnn}\U{nnnnnnnn}scanread.tableallowEscapes = TRUE\nnn\377\xff
\U\U10FFFF
```

```
print.default
```

```
\0
```

```
r"(...)..."[]{}Rr
```

```
.
```

```
`deparseformula
```

```
\unnnn\uoooo\uD834\uDD1E\u1D11E"abc\uD834de"
```

```
Syntax
```

```
sQuote
```

```
shQuote
```

```
'single quotes can be used more-or-less interchangeably'
"with double quotes to create character vectors"
```

```
## Single quotes inside single-quoted strings need backslash-escaping.
## Ditto double quotes inside double-quoted strings.
##
identical('"It\'s alive!", he screamed.',
          "\"It's alive!\", he screamed.") # same
```

```
## Backslashes need doubling, or they have a special meaning.
x <- "In ALGOL, you could do logical AND with /\\"
print(x)      # shows it as above ("input-like")
writeLines(x) # shows it as you like it ;-)
```

```
## Single backslashes followed by a letter are used to denote
## special characters like tab(ulator)s and newlines:
x <- "long\tlines can be\nbroken with newlines"
writeLines(x) # see also ?strwrap
```

```
## Backticks are used for non-standard variable names.
## (See make.names and ?Reserved for what counts as
## non-standard.)
`x` `y` <- 1:5
```



```

`x y`
d <- data.frame(`1st column` = rchisq(5, 2), check.names = FALSE)
d$`1st column`

## Backslashes followed by up to three numbers are interpreted as
## octal notation for ASCII characters.
"\110\145\154\154\157\40\127\157\162\154\144\41"

## \x followed by up to two numbers is interpreted as
## hexadecimal notation for ASCII characters.
(hw1 <- "\x48\x65\x6c\x6c\x6f\x20\x57\x6f\x72\x6c\x64\x21")

## Mixing octal and hexadecimal in the same string is OK
(hw2 <- "\110\x65\154\x6c\157\x20\127\x6f\162\x6c\144\x21")

## \u is also hexadecimal, but supports up to 4 digits,
## using Unicode specification. In the previous example,
## you can simply replace \x with \u.
(hw3 <- "\u48\u65\u6c\u6c\u6f\u20\u57\u6f\u72\u6c\u64\u21")

## The last three are all identical to
hw <- "Hello World!"
stopifnot(identical(hw, hw1), identical(hw1, hw2), identical(hw2, hw3))

## Using Unicode makes more sense for non-latin characters.
(nn <- "\u0126\u0119\u1114\u022d\u2001\u03e2\u0954\u0f3f\u13d3\u147b\u203c")

## Mixing \x and \u throws a _parse_ error (which is not catchable!)
## Not run:
"\x48\u65\x6c\u6c\x6f\u20\u57\u6f\u72\u6c\u64\u21"

## End(Not run)
## --> Error: mixing Unicode and octal/hex escapes .....

## \U works like \u, but supports up to six hex digits.
## So we can replace \u with \U in the previous example.
n2 <- "\U0126\U0119\U1114\U022d\U2001\U03e2\U0954\U0f3f\U13d3\U147b\U203c"
stopifnot(identical(nn, n2))

## Under systems supporting multi-byte locales (and not Windows),
## \U also supports the rarer characters outside the usual 16^4 range.
## See the R language manual,
## https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Literal-constants
## and bug 16098 https://bugs.r-project.org/show\_bug.cgi?id=16098
## This character may or not be printable (the platform decides)
## and if it is, may not have a glyph in the font used.
"\U1d4d7" # On Windows this used to give the incorrect value of "\Ud4d7"

## nul characters (for terminating strings in C) are not allowed (parse errors)
## Not run:
"foo\0bar" # Error: nul character not allowed (line 1)
"foo\u0000bar" # same error

## End(Not run)

## A Windows path written as a raw string constant:
r"(c:\Program files\R)"

```

```
## More raw strings:
r"{{\1\2}}"
r"(use both "double" and 'single' quotes)"
r"---(\1--)----"
```

R.Version

```
R.Version()
R.versionlistversion
```

```
R.Version()
R.version
R.version.string
version
```

```
R_compiled_by()
```

[Sys.info](#)

```
darwin13.3.0linux-gnusolaris2.10mingw32
R.version$crt"ucrt""msvcrt"
```

```
R.Version
```

```
platform      "i686-unknown-linux-gnu""i386-pc-mingw32"
arch
os
crt
system
status        "alpha"
major
minor
year
month
day
svn rev       "unknown"MS
language      "R"
version.string character
```

```
R.versionversion"simple.list"print
R_compiled_by
```

```
R.version$os.Platform$OS.typeR.version$ososVersion
R.version.stringR.version$version.string
```

```
sessionInfogetRversionosVersion.PlatformSys.info
```

```
require(graphics)

R.version$os # to check how lucky you are ...
plot(0) # any plot
mtext(R.version.string, side = 1, line = 4, adj = 1) # a useful bottom-right note

## a good way to detect macOS:
if(grepl("^darwin", R.version$os)) message("running on macOS")

## Short R version string, ("space free", useful in file/directory names;
## also fine for unreleased versions of R):
shortRversion <- function() {
  rvs <- R.version.string
  if(grepl("devel", (st <- R.version$status)))
    rvs <- sub(paste0(" ",st," "), "-devel_", rvs, fixed=TRUE)
  gsub("[()]", "", gsub(" ", "_", sub(" version ", "-", rvs)))
}
shortRversion()
```

Random

```
.Random.seed
RNGkind
RNGversion
set.seed

.Random.seed <- c(rng.kind, n1, n2, ...)

RNGkind(kind = NULL, normal.kind = NULL, sample.kind = NULL)
RNGversion(vstr)
set.seed(seed, kind = NULL, normal.kind = NULL, sample.kind = NULL)

kind          NULLkind"default"NULL
normal.kind    NULL"default"NULL
sample.kind    NULLsample"default"NULL
seed           NULL
vstr           "1.6.2"vstr
rng.kind       0:kkind
n1n2...        rng.kind
```

```

kind"Mersenne-Twister"

"Wichmann-Hill" .Random.seed[-1] == r[1:3]r[i]1:(p[i] - 1)pp = (30269, 30307,
    30323)6.9536 × 1012prod(p-1)/4
"Marsaglia-Multicarry" sci.stat.math260

"Super-Duper" ≈ 4.6 × 1018

"Mersenne-Twister" 219937 - 1

"Knuth-TAOCP-2002"

$$X_j = (X_{j-100} - X_{j-37}) \bmod 2^{30}$$

2129
"Knuth-TAOCP"

"L'Ecuyer-CMRG" 2191
42949670874294944443

"user-supplied" Random.user

normal.kind"Kinderman-Ramage""Buggy" Kinderman-Ramage"set.seed"Ahrens-Dieter"
"Box-Muller""Inversion""user-supplied"qnorm"Buggy""Box-Muller"kind
sample.kind"Rounding""Rejection"sample
set.seed"Mersenne-Twister""Knuth-TAOCP"seedseed = NULL
kind = NULLnormal.kind = NULLsample.kind = NULLRNGkindset.seed"default"

.Random.seedinteger0:(k-1)k0
.Random.seed[-1]unsigned.Random.seed[-1]
RNGkindNULLRNGkind.Random.seed
RNGversionRNGkind
set.seedNULL

.Random.seedset.seedkindnormal.kind.Random.seed
.Random.seed
232

```

set.seed.Random.seed

<https://www-cs-faculty.stanford.edu/~knuth/taocp.html>

<https://simul.iro.umontreal.ca/testu01/tu01.html><https://github.com/umontreal-simul/TestU01-2009>

sci.stat.math

<http://www.math.keio.ac.jp/~matumoto/emt.html>

<https://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/VERSIONS/C-LANG/c-lang.html>

sample

```
require(stats)
```

```
## Seed the current RNG, i.e., set the RNG status
```

```
set.seed(42); u1 <- runif(30)
```

```
set.seed(42); u2 <- runif(30) # the same because of identical RNG status:
```

```
stopifnot(identical(u1, u2))
```

```
## the default random seed is 626 integers, so only print a few
```

```
runif(1); .Random.seed[1:6]; runif(1); .Random.seed[1:6]
```

```
## If there is no seed, a "random" new one is created:
```

```
rm(.Random.seed); runif(1); .Random.seed[1:6]
```

```
ok <- RNGkind()
```

```
RNGkind("Wich") # (partial string matching on 'kind')
```

```
## This shows how 'runif(.)' works for Wichmann-Hill,
```

```
## using only R functions:
```

```
p.WH <- c(30269, 30307, 30323)
```

```

a.WH <- c( 171, 172, 170)
next.WHseed <- function(i.seed = .Random.seed[-1])
  { (a.WH * i.seed) %% p.WH }
my.runif1 <- function(i.seed = .Random.seed)
  { ns <- next.WHseed(i.seed[-1]); sum(ns / p.WH) %% 1 }
set.seed(1998-12-04)# (when the next lines were added to the source)
rs <- .Random.seed
(WHs <- next.WHseed(rs[-1]))
u <- runif(1)
stopifnot(
  next.WHseed(rs[-1]) == .Random.seed[-1],
  all.equal(u, my.runif1(rs))
)

## ----
.Random.seed
RNGkind("Super") # matches "Super-Duper"
RNGkind()
.Random.seed # new, corresponding to Super-Duper

## Reset:
RNGkind(ok[1])

RNGversion(getRversion()) # the default version for this R version

## ----
sum(duplicated(runif(1e6))) # around 110 for default generator
## and we would expect about almost sure duplicates beyond about
qbirthday(1 - 1e-6, classes = 2e9) # 235,000

```

Random.user

[RNGkind](#)

```

user_unif_rand
user_unif_initunsigned intRNGkindset.seedseedset.seedRNGkind
.Random.seeduser_unif_nseeduser_unif_seedlocInt32GetRNGstatePutRNGstate
.Random.seed
user_norm_rand

```

R_ext/Random.h

```

## Not run:
## Marsaglia's congruential PRNG
#include <R_ext/Random.h>

static Int32 seed;
static double res;

```

```

static int nseed = 1;

double * user_unif_rand(void)
{
    seed = 69069 * seed + 1;
    res = seed * 2.32830643653869e-10;
    return &res;
}

void user_unif_init(Int32 seed_in) { seed = seed_in; }
int * user_unif_nseed(void) { return &nseed; }
int * user_unif_seedloc(void) { return (int *) &seed; }

/* ratio-of-uniforms for normal */
#include <math.h>
static double x;

double * user_norm_rand(void)
{
    double u, v, z;
    do {
        u = unif_rand();
        v = 0.857764 * (2. * unif_rand() - 1);
        x = v/u; z = 0.25 * x * x;
        if (z < 1. - u) break;
        if (z > 0.259/u + 0.35) continue;
    } while (z > -log(u));
    return &x;
}

## Use under Unix:
R CMD SHLIB urand.c
R
> dyn.load("urand.so")
> RNGkind("user")
> runif(10)
> .Random.seed
> RNGkind("user")
> rnorm(10)
> RNGkind()
[1] "user-supplied" "user-supplied"

## End(Not run)

```

range

range

```

range(..., na.rm = FALSE)
## Default S3 method:
range(..., na.rm = FALSE, finite = FALSE)
## same for classes 'Date' and 'POSIXct'

.rangeNum(..., na.rm, finite, isNumeric)

```

```

...          numeric
na.rm        NA
finite
isNumeric    functionTRUEFALSEc(..., recursive = TRUE)is.numeric()range()

```

```

rangeSummary...
na.rmFALSENaNNaNNANA
finiteTRUEfinite = TRUEna.rm = TRUE
NAmin

```

```

Summaryx, ..., na.rm

```

```

minmax
extendrange()

```

```

(r.x <- range(stats::rnorm(100)))
diff(r.x) # the SAMPLE range

x <- c(NA, 1:3, -1:1/0); x
range(x)
range(x, na.rm = TRUE)
range(x, finite = TRUE)

```

rank

```

rank(x, na.last = TRUE,
      ties.method = c("average", "first", "last", "random", "max", "min"))

```

```

x
na.last      NATRUEFALSENA"keep"NA
ties.method

```



```

NAseq_along(x)ties.method"first""last""random""average""max""min"
NAAna.last = TRUEAna.last = FALSEx
rankxtfrmrank(xtfrm(x), ...)xtfrmrank==>is.na

xxna.last = NAties.method = "average"

```

ordersortxtfrm

```

(r1 <- rank(x1 <- c(3, 1, 4, 15, 92)))
x2 <- c(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5)
names(x2) <- letters[1:11]
(r2 <- rank(x2)) # ties are averaged

## rank() is "idempotent": rank(rank(x)) == rank(x) :
stopifnot(rank(r1) == r1, rank(r2) == r2)

## ranks without averaging
rank(x2, ties.method= "first") # first occurrence wins
rank(x2, ties.method= "last")  # last occurrence wins
rank(x2, ties.method= "random") # ties broken at random
rank(x2, ties.method= "random") # and again

## keep ties ties, no average
(rma <- rank(x2, ties.method= "max")) # as used classically
(rmi <- rank(x2, ties.method= "min")) # as in Sports
stopifnot(rma + rmi == round(r2 + r2))

## Comparing all tie.methods:
tMeth <- eval(formals(rank)$ties.method)
rx2 <- sapply(tMeth, function(M) rank(x2, ties.method=M))
cbind(x2, rx2)
## ties.method's does not matter w/o ties:
x <- sample(47)
rx <- sapply(tMeth, function(MM) rank(x, ties.method=MM))
stopifnot(all(rx[,1] == rx))

```

rapply

```

rapplylapplyhow = ".."

```

```

rapply(object, f, classes = "ANY", deflt = NULL,
        how = c("unlist", "replace", "list"), ...)

```

```

object      listexpression
f           function...
classes     class"ANY"
deflt       how = "replace"
how
...         f

```

```

how = "replace"objectclassesf
how = "list"how = "unlist"objectclassesfdeflthow = "unlist"unlist(recursive =
TRUE)

```

```

lapply
rapply()lapply()f
object

```

```

how = "unlist"object

```

```

rapply

```

```

lapplydendrapply

```

```

X <- list(list(a = pi, b = list(c = 1L)), d = "a test")
# the "identity operation":
rapply(X, function(x) x, how = "replace") -> X.; stopifnot(identical(X, X.))
rapply(X, sqrt, classes = "numeric", how = "replace")
rapply(X, deparse, control = "all") # passing extras. argument of deparse()
rapply(X, nchar, classes = "character", deflt = NA_integer_, how = "list")
rapply(X, nchar, classes = "character", deflt = NA_integer_, how = "unlist")
rapply(X, nchar, classes = "character", how = "unlist")
rapply(X, log, classes = "numeric", how = "replace", base = 2)

## with expression() / list():
E <- expression(list(a = pi, b = expression(c = C1 * C2)), d = "a test")
LE <- list(expression(a = pi, b = expression(c = C1 * C2)), d = "a test")
rapply(E, nchar, how="replace") # "expression(c = C1 * C2)" are 23 chars
rapply(E, nchar, classes = "character", deflt = NA_integer_, how = "unlist")
rapply(LE, as.character) # a "pi" | b1 "expression" | b2 "C1 * C2" ..
rapply(LE, nchar) # (see above)
stopifnot(exprs = {
  identical(E , rapply(E , identity, how = "replace"))
  identical(LE, rapply(LE, identity, how = "replace"))
})

```

raw

"raw"

```
raw(length = 0)
as.raw(x)
is.raw(x)
```

```
length
x
```

[rawToChar](#)

```
[0 ... 255]NA0nul
as.rawis.raw
as.raw
```

```
raw0
as.raw0
is.rawtypeof(x) == "raw"
```

[charToRawrawShift](#)

[&](#)

```
xx <- raw(2)
xx[1] <- as.raw(40)      # NB, not just 40.
xx[2] <- charToRaw("A")
xx      ## 28 41  -- raw prints hexadecimal
dput(xx) ## as.raw(c(0x28, 0x41))
as.integer(xx) ## 40 65
```

```
x <- "A test string"
(y <- charToRaw(x))
is.vector(y) # TRUE
rawToChar(y)
is.raw(x)
is.raw(y)
stopifnot( charToRaw("\xa3") == as.raw(0xa3) )
```

```
isASCII <- function(txt) all(charToRaw(txt) <= as.raw(127))
isASCII(x) # true
isASCII("\xa325.63") # false (in Latin-1, this is an amount in UK pounds)
```

rawConnection

rawConnection(object, open = "r")

rawConnectionValue(con)

object

open

con

close

rawConnectionValue

rawConnection"rawConnection""connection"

rawConnectionValue

[file\(\)](#)

vsnprintf

[connectionsshowConnections](#)

```
zz <- rawConnection(raw(0), "r+") # start with empty raw vector
writeBin(LETTERS, zz)
seek(zz, 0)
readLines(zz) # raw vector has embedded nuls
seek(zz, 0)
writeBin(letters[1:3], zz)
rawConnectionValue(zz)
close(zz)
```

rawConversion

"raw"

charToRaw(x)
rawToChar(x, multiple = FALSE)

rawShift(x, n)

rawToBits(x)
intToBits(x)
packBits(x, type = c("raw", "integer", "double"))

numToInts(x)
numToBits(x)

x
multiple
n -8 ... 8
type

packBits
numToBits(.)packBits(., type="double")

charToRaw[Encoding](#)
rawToChar""0multiple
rawShift(x, n)xnn
rawToBitsintToBits
packBits
numToInts()numToBits()[doubleinteger](#)raw

```
x <- "A test string"
(y <- charToRaw(x))
is.vector(y) # TRUE
```

```
rawToChar(y)
rawToChar(y, multiple = TRUE)
(xx <- c(y, charToRaw("&"), charToRaw(" more")))
rawToChar(xx)
```

```

rawShift(y, 1)
rawShift(y,-2)

rawToBits(y)

showBits <- function(r) stats::symnum(as.logical(rawToBits(r)))

z <- as.raw(5)
z ; showBits(z)
showBits(rawShift(z, 1)) # shift to right
showBits(rawShift(z, 2))
showBits(z)
showBits(rawShift(z, -1)) # shift to left
showBits(rawShift(z, -2)) # ..
showBits(rawShift(z, -3)) # shifted off entirely

packBits(as.raw(0:31))
i <- -2:3
stopifnot(exprs = {
  identical(i, packBits(intToBits(i), "integer"))
  identical(packBits(      0:31) ,
            packBits(as.raw(0:31)))
})
str(pBi <- packBits(intToBits(i)))
data.frame(B = matrix(pBi, nrow=6, byrow=TRUE),
           hex = format(as.hexmode(i)), i)

## Look at internal bit representation of ...

## ... of integers :
bitI <- function(x) vapply(as.integer(x), function(x) {
  b <- substr(as.character(rev(intToBits(x))), 2L, 2L)
  paste0(c(b[1L], " ", b[2:32]), collapse = "")
}, "")
print(bitI(-8:8), width = 35, quote = FALSE)

## ... of double precision numbers in format 'sign exp | mantissa'
## where 1 bit sign 1 <==> "-";
##      11 bit exp   is the base-2 exponent biased by 2^10 - 1 (1023)
##      52 bit mantissa is without the implicit leading '1'
#
## Bit representation [ sign | exponent | mantissa ] of double prec numbers :

bitC <- function(x) noquote(vapply(as.double(x), function(x) { # split one double
  b <- substr(as.character(rev(numToBits(x))), 2L, 2L)
  paste0(c(b[1L], " ", b[2:12], " | ", b[13:64]), collapse = "")
}, ""))
bitC(17)
bitC(c(-1,0,1))
bitC(2^(-2:5))
bitC(1+2^-(1:53))# from 0.5 converge to 1

### numToBits(.) <==> intToBits(numToInts(.)) :
d2bI <- function(x) vapply(as.double(x), function(x) intToBits(numToInts(x)), raw(64L))
d2b <- function(x) vapply(as.double(x), function(x)      numToBits(x) , raw(64L))
set.seed(1)

```

```

x <- c(sort(rt(2048, df=1.5)), 2^(-10:10), 1+2^-(1:53))
str(bx <- d2b(x)) # a 64 x 2122 raw matrix
stopifnot( identical(bx, d2bI(x)) )

## Show that packBits(*, "double") is the inverse of numToBits() :
packBits(numToBits(pi), type="double")
bitC(2050)
b <- numToBits(2050)
identical(b, numToBits(packBits(b, type="double")))
pbx <- apply(bx, 2, packBits, type="double")
stopifnot( identical(pbx, x) )

```

RdUtils

R CMD Rdconv [options] file
 R CMD Rd2pdf [options] files

file
 files
 options

R CMD Rdconv
 R CMD Rd2pdfR_PAPERSIZER CMDR_PAPERSIZEa4letterlegalexecutiveR_PDFVIEWER
 RD2PDF_INPUTENCinputenxinputenc
 R CMD Rd2pdftools::texi2pdftexi2dviR_TEXI2DVICMD
 R CMD --help

[RShowDoc](#)("R-exts")

readBin

```

readBin(con, what, n = 1L, size = NA_integer_, signed = TRUE,
        endian = .Platform$endian)

writeBin(object, con, size = NA_integer_,
        endian = .Platform$endian, useBytes = FALSE)

```

```

con
what      "numeric""double""integer""int""logical""complex""character"
          "raw"
n          n
size      NA_integer_
signed
endian    "big""little""swap"
object
useBytes  writeLines

```

```

confile

```

```

readBinconwriteBincon
size signed = FALSE NA
readBinwriteBinreadCharwriteChar
Inf-InfNaN
 $2^{31} - 1$ 
size > 1

```

```

readBinn
writeBinconNULL

```

```

longlong long
sizeof(long double)double
readBin(what = character())

```

```

readChar
connectionsreadLineswriteLines
.Machinelonglong longlong double

```

```

zzfil <- tempfile("testbin")
zz <- file(zzfil, "wb")
writeBin(1:10, zz)
writeBin(pi, zz, endian = "swap")
writeBin(pi, zz, size = 4)
writeBin(pi^2, zz, size = 4, endian = "swap")
writeBin(pi+3i, zz)
writeBin("A test of a connection", zz)

```



```

z <- paste("A very long string", 1:100, collapse = " + ")
writeBin(z, zz)
if(.Machine$sizeof.long == 8 || .Machine$sizeof.longlong == 8)
  writeBin(as.integer(5^(1:10)), zz, size = 8)
if((s <- .Machine$sizeof.longdouble) > 8)
  writeBin((pi/3)^(1:10), zz, size = s)
close(zz)

zz <- file(zzfil, "rb")
readBin(zz, integer(), 4)
readBin(zz, integer(), 6)
readBin(zz, numeric(), 1, endian = "swap")
readBin(zz, numeric(), size = 4)
readBin(zz, numeric(), size = 4, endian = "swap")
readBin(zz, complex(), 1)
readBin(zz, character(), 1)
z2 <- readBin(zz, character(), 1)
if(.Machine$sizeof.long == 8 || .Machine$sizeof.longlong == 8)
  readBin(zz, integer(), 10, size = 8)
if((s <- .Machine$sizeof.longdouble) > 8)
  readBin(zz, numeric(), 10, size = s)
close(zz)
unlink(zzfil)
stopifnot(z2 == z)

## signed vs unsigned ints
zzfil <- tempfile("testbin")
zz <- file(zzfil, "wb")
x <- as.integer(seq(0, 255, 32))
writeBin(x, zz, size = 1)
writeBin(x, zz, size = 1)
x <- as.integer(seq(0, 60000, 10000))
writeBin(x, zz, size = 2)
writeBin(x, zz, size = 2)
close(zz)
zz <- file(zzfil, "rb")
readBin(zz, integer(), 8, size = 1)
readBin(zz, integer(), 8, size = 1, signed = FALSE)
readBin(zz, integer(), 7, size = 2)
readBin(zz, integer(), 7, size = 2, signed = FALSE)
close(zz)
unlink(zzfil)

## use of raw
z <- writeBin(pi^{1:5}, raw(), size = 4)
readBin(z, numeric(), 5, size = 4)
z <- writeBin(c("a", "test", "of", "character"), raw())
readBin(z, character(), 4)

```

readChar

```
readChar(con, nchars, useBytes = FALSE)
```

```
writeChar(object, con, nchars = nchar(object, type = "chars"),  
          eos = "", useBytes = FALSE)
```

con	
nchars	NA
useBytes	readCharncharswriteCharwriteLines
object	nchars
eos	nuNULL

```
readBinwriteBin  
confile
```

```
readCharconwriteCharcon  
nulreadCharnul  
readCharwriteChar  
NA
```

```
readCharlength(nchars)  
writeCharconNULL
```

```
readCharreadBinreadCharreadBin(what = "raw")  
nchars
```

```
encoding
```

```
connectionsreadLineswriteLinesreadBin
```

```
## test fixed-length strings  
zzfil <- tempfile("testchar")  
zz <- file(zzfil, "wb")  
x <- c("a", "this will be truncated", "abc")  
nc <- c(3, 10, 3)  
writeChar(x, zz, nc, eos = NULL)  
writeChar(x, zz, eos = "\r\n")  
close(zz)  
  
zz <- file(zzfil, "rb")  
readChar(zz, nc)  
readChar(zz, nchar(x)+3) # need to read the terminator explicitly  
close(zz)  
unlink(zzfil)
```

readline

readline

readline(prompt = "")

prompt " "

""

readLines

```
fun <- function() {  
  ANSWER <- readline("Are you a satisfied R user? ")  
  ## a better version would check the answer less cursorily, and  
  ## perhaps re-prompt  
  if (substr(ANSWER, 1, 1) == "n")  
    cat("This is impossible. YOU LIED!\n")  
  else  
    cat("I knew it.\n")  
}  
if(interactive()) fun()
```

readLines

readLines(con = stdin(), n = -1L, ok = TRUE, warn = TRUE,
 encoding = "unknown", skipNul = FALSE)

con

n

ok n > 0

warn

encoding conoptions(encoding=)

skipNul

```
confilefile
```

```
"rt"close
```

```
skipNul = TRUEwarn = FALSE
```

```
conencodingencodingreadLines
```

```
encoding"latin1""UTF-8"
```

```
stdincon = "stdin"file
```

```
connectionswriteLinesreadBinscan
```

```
fil <- tempfile(fileext = ".data")
cat("TITLE extra line", "2 3 5 7", "", "11 13 17", file = fil,
    sep = "\n")
readLines(fil, n = -1)
unlink(fil) # tidy up

## difference in blocking
fil <- tempfile("test")
cat("123\nabc", file = fil)
readLines(fil) # line with a warning

con <- file(fil, "r", blocking = FALSE)
readLines(con) # "123"
cat(" def\n", file = fil, append = TRUE)
readLines(con) # gets both
close(con)

unlink(fil) # tidy up

## Not run:
# read a 'Windows Unicode' file
A <- readLines(con <- file("Unicode.txt", encoding = "UCS-2LE"))
close(con)
unique(Encoding(A)) # will most likely be UTF-8

## End(Not run)
```

readRDS

```
saveRDS(object, file = "", ascii = FALSE, version = NULL,  
        compress = TRUE, refhook = NULL)
```

```
readRDS(file, refhook = NULL)  
infoRDS(file)
```

object

file

ascii TRUE [NA](#) [save](#)

version NULL

compress "gzip" "gzip" "bzip2" "xz" "zstd" file

refhook

```
saveRDSreadRDSsaveloadhelp.search".rds"
```

```
serializeunserializeserializereadRDS
```

```
infoRDSsaveRDSserializeinfoRDS
```

```
save"RDXs\n"
```

```
filegzfilesave(compress = FALSE)filefile
```

```
filefileurlgzcon
```

```
saveRDS(ascii = FALSE)
```

readRDS

saveRDSNULL

```
infoRDSversionwriter_versionmin_reader_versionformatnative_encoding"xdr""ascii"
```

```
ascii = TRUEascii = NA"binary"serialize
```

```
saveRDSserializesave
```

```
compress = "zstd"
```

```
serializesaveload
```

```

fil <- tempfile("women", fileext = ".rds")
## save a single object to file
saveRDS(women, fil)
## restore it under a different name
women2 <- readRDS(fil)
identical(women, women2)
## or examine the object via a connection, which will be opened as needed.
con <- gzfile(fil)
readRDS(con)
close(con)

## Less convenient ways to restore the object
## which demonstrate compatibility with unserialize()
con <- gzfile(fil, "rb")
identical(unserialize(con), women)
close(con)
con <- gzfile(fil, "rb")
wm <- readBin(con, "raw", n = 1e4) # size is a guess
close(con)
identical(unserialize(wm), women)

## Format compatibility with serialize():
fil2 <- tempfile("women")
con <- file(fil2, "w")
serialize(women, con) # ASCII, uncompressed
close(con)
identical(women, readRDS(fil2))
fil3 <- tempfile("women")
con <- bzfile(fil3, "w")
serialize(women, con) # binary, bzip2-compressed
close(con)
identical(women, readRDS(fil3))

unlink(c(fil, fil2, fil3))

```

readRenviron

.RenvironRenviron.site

readRenviron(path)

path

[Startup](#)

```
## Not run:  
## re-read a startup file (or read it in a vanilla session)  
readRenviroN("~/RenviroN")  
  
## End(Not run)
```

Recall

Recall

Recall(...)

...

Recallapply

do.callcall

local

```
## A trivial (but inefficient!) example:  
fib <- function(n)  
  if(n<=2) { if(n>=0) 1 else 0 } else Recall(n-1) + Recall(n-2)  
fibonacci <- fib; rm(fib)  
## renaming wouldn't work without Recall  
fibonacci(10) # 55
```

reg.finalizer

reg.finalizer(e, f, onexit = FALSE)

e

f

onexit

[.Last](#)

NULL

[foptions](#)

[gcMemory](#)

```
f <- function(e) print("cleaning...")
g <- function(x){ e <- environment(); reg.finalizer(e, f) }
g()
invisible(gc()) # trigger cleanup
```

regex

[grep](#)[grepl](#)[regexpr](#)[gregexpr](#)[subgs](#)[substr](#)[split](#)[tag](#)[rep](#)[agrep](#)[repl](#)

```
perl = TRUEfixed = TRUE
grepaproposbrowseEnvhelp.searchlist.filesls
cat
```

```
grepgreplregexprgregexprsubgsregexecstrsplit
useBytes = TRUE
. \ | ( ) [ { ^ $ * + ?
\aBEL\eESC\fFF\nLF\rCR\tTAB
[]^[0123456789][^abc]abc
[ABCDEFGHIIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz]
```

```
[:alnum:] [:alpha:] [:digit:]
[:alpha:] [:lower:] [:upper:]
[:blank:]
[:cntrl:] DEL
[:digit:] 0 1 2 3 4 5 6 7 8 9
```



```
[:graph:] [:alnum:][:punct:]
[:lower:]
[:print:] [:alnum:][:punct:]
[:punct:]
    ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
[:space:]
[:upper:]
[:xdigit:]
    0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

[[:alnum:]] [0-9A-Za-z]]^~perl = TRUE^ - \ ]
.\w[[:alnum:]]_]\W[^[:alnum:]]_]\d\s\D\S
^$\<\>\b\B
```

```
?
*
+
{n} n
{n,} n
{n,m} nm
?
```

```
|abba|cdeabbacde|
```

```
\NN = 1 ... 9
```

```
perl = TRUEgrepregexprgrepregexprsubgsubstrsplit
man pcrepatternman pcreapihttps://www.pcre.org/nextSoftVersion
https://www.pcre.org/original/doc/html/https://www.pcre.org/current/doc/html/
Encoding
\<\>{sub
[A-Za-z]
\p
(?...)?
(?i)/i(?m)/m(?s)/s(?x)/x(?im)(?im-sx)(?U)?
\Q\E$@Q...E$@
\d\s\wC\h\v\H\V
\cxctr1-xx\ddd\1\7\hhh\h{h...}
\A^\Z\z\G\C\R\X\X\R\B\XRB
\-
```

```
\p{xx}\P{xx}xxLuSc\w\W\d\D\s\S\b\B(*UCP)pcre_config
(?#
#
(?:...)
(?:=...)(?!...)...(?<=...)(?<!...)\C...
regexprregexpr"(?<first>[A-Z][a-z]+)"sub
```

pcre2pattern

[grepaproposbrowseEnvglob2rxhelp.searchlist.fileslsstrsplitagrep](#)

https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html
pcre2patternpcrpatternman<https://www.pcre.org/original/pcre.txt><https://perldoc.perl.org/perlre>

regmatches

[regexprgregexprregexecgregexec](#)

```
regmatches(x, m, invert = FALSE)
regmatches(x, m, invert = FALSE) <- value
```

```
x
m
invert      TRUE
value      Details
```

```
invertFALSEregmatchesregexpr
invertTRUEregmatchesstrsplit
invertNAregmatches
x
invertFALSEvaluemm
```

```
regmatchesminvertFALSE
regmatches<-
```

```

x <- c("A and B", "A, B and C", "A, B, C and D", "foobar")
pattern <- "[[:space:]]*(,|and)[[:space:]]"
## Match data from regexr()
m <- regexr(pattern, x)
regmatches(x, m)
regmatches(x, m, invert = TRUE)
## Match data from gregexpr()
m <- gregexpr(pattern, x)
regmatches(x, m)
regmatches(x, m, invert = TRUE)

## Consider
x <- "John (fishing, hunting), Paul (hiking, biking)"
## Suppose we want to split at the comma (plus spaces) between the
## persons, but not at the commas in the parenthesized hobby lists.
## One idea is to "blank out" the parenthesized parts to match the
## parts to be used for splitting, and extract the persons as the
## non-matched parts.
## First, match the parenthesized hobby lists.
m <- gregexpr("\\([^\)]*\\)", x)
## Create blank strings with given numbers of characters.
blanks <- function(n) strrep(" ", n)
## Create a copy of x with the parenthesized parts blanked out.
s <- x
regmatches(s, m) <- Map(blanks, lapply(regmatches(s, m), nchar))
s
## Compute the positions of the split matches (note that we cannot call
## strsplit() on x with match data from s).
m <- gregexpr(" ", s)
## And finally extract the non-matched parts.
regmatches(x, m, invert = TRUE)

## regexec() and gregexec() return overlapping ranges because the
## first match is the full match. This conflicts with regmatches()<-
## and regmatches(..., invert=TRUE). We can work-around by dropping
## the first match.
drop_first <- function(x) {
  if(!anyNA(x) && all(x > 0)) {
    ml <- attr(x, 'match.length')
    if(is.matrix(x)) x <- x[-1,] else x <- x[-1]
    attr(x, 'match.length') <- if(is.matrix(ml)) ml[-1,] else ml[-1]
  }
  x
}
m <- gregexec("(\\w+) \\(((?:\\w+(?:, )?)+)\\)", x)
regmatches(x, m)
try(regmatches(x, m, invert=TRUE))
regmatches(x, lapply(m, drop_first))
## invert=TRUE loses matrix structure because we are retrieving what
## is in between every sub-match
regmatches(x, lapply(m, drop_first), invert=TRUE)
y <- z <- x
## Notice **list**(...) on the RHS
regmatches(y, lapply(m, drop_first)) <- list(c("<NAME>", "<HOBBY-LIST>"))
y
regmatches(z, lapply(m, drop_first), invert=TRUE) <-

```

```

    list(sprintf("<%d>", 1:5))
z

## With `perl = TRUE` and `invert = FALSE` capture group names
## are preserved. Collect functions and arguments in calls:
NEWS <- head(readLines(file.path(R.home("doc"), "NEWS.2")), 100)
m <- grexexec("(?<fun>\\w+)\\((?<args>[^]*)\\)", NEWS, perl = TRUE)
y <- regmatches(NEWS, m)
y[[16]]
## Make tabular, adding original line numbers
mdat <- as.data.frame(t(do.call(cbind, y)))
mdat <- cbind(mdat, line=rep(seq_along(y), lengths(y) / ncol(mdat)))
head(mdat)
NEWS[head(mdat[['line']])]

```

remove

```

removermlist
envir
inheritsTRUE

```

```

remove(..., list = character(), pos = -1,
        envir = as.environment(pos), inherits = FALSE)

rm      (... , list = character(), pos = -1,
        envir = as.environment(pos), inherits = FALSE)

```

```

...
list          NULL
pos
envir         environment
inherits

```

```

possearchenvironmentsys.frameenvir
lockEnvironment
...list

```

[lsobjects](#)

```

tmp <- 1:4
## work with tmp and cleanup
rm(tmp)

## Not run:
## remove (almost) everything in the working environment.
## You will get no warning, so don't do this unless you are really sure.
rm(list = ls())

## End(Not run)

```

rep

```

repx
rep.intrep_len

```

```

rep(x, ...)
rep.int(x, times)
rep_len(x, length.out)

```

```

x          listrepPOSIXctPOSIXltDate
...
          times length(x)NAdouble
          length.out NA
          each xeach1NA
timeslength.out
          ...

```

```

rep(x, times = 1, length.out = NA, each = 1)

eachtimeslength.out
timetimesxeachx[1]times[1]x[2]times[2]
length.outtimesxlength.outtimes
timetimesroundeach
xlength.outNA0NULL

```

```

x
rep.intrep_len
repx

```

```
rep.intrep.intx
rep.intrep
rep
repNULLNULLlength.out
```

seqsequencereplicate

```
rep(1:4, 2)
rep(1:4, each = 2)      # not the same.
rep(1:4, c(2,2,2,2))    # same as second.
rep(1:4, c(2,1,2,1))
rep(1:4, each = 2, length.out = 4)    # first 4 only.
rep(1:4, each = 2, length.out = 10)    # 8 integers plus two recycled 1's.
rep(1:4, each = 2, times = 3)          # length 24, 3 complete replications

rep(1, 40*(1-.8)) # length 7 on most platforms
rep(1, 40*(1-.8)+1e-7) # better

## replicate a list
fred <- list(happy = 1:10, name = "squash")
rep(fred, 5)

# date-time objects
x <- .leap.seconds[1:3]
rep(x, 2)
rep(as.POSIXlt(x), rep(2, 3))

## named factor
x <- factor(LETTERS[1:4]); names(x) <- letters[1:4]
x
rep(x, 2)
rep(x, each = 2)
rep.int(x, 2) # no names
rep_len(x, 10)
```

replace

```
replacexlistvaluesvalues
```

```
replace(x, list, values)
```

x
list
values

x

Reserved

ifelse repeat while function for in next break
TRUE FALSE NULL Inf NaN NA integer NA real NA complex NA character_
.....1..2...

make.names

rev

rev dendrogram
sort(x, decreasing = TRUE) rev(sort(x))

rev(x)

x

seqsort

```
x <- c(1:5, 5:3)
## sort into descending order; first more efficiently:
stopifnot(sort(x, decreasing = TRUE) == rev(sort(x)))
stopifnot(rev(1:7) == 7:1) #- don't need 'rev' here
```

Rhome

```
R.home(component = "home")
```

```
component      "home""bin""doc""etc""include""modules""share"
```

```
R RHOME
```

```
"doc""include""share"
```

```
"modules""bin""etc"
```

```
R_HOMER_SHARE_DIRR_DOC_DIRR_INCLUDE_DIR
```

```
R.home()R_HOMER_HOMEMakefile
```

```
commandArgs()[1]
```

```
## These result quite platform-dependently :
```

```
rbind(home = R.home(),  
      bin  = R.home("bin")) # often the 'bin' sub directory of 'home'  
                           # but not always ...
```

```
list.files(R.home("bin"))
```

rle

```
rle(x)  
inverse.rle(x, ...)
```

```
## S3 method for class 'rle'
```

```
print(x, digits = getOption("digits"), prefix = "", ...)
```



```

x          rle()"rle"inverse.rle()
...
digits     print.default
prefix

is.vector

inverse.rle()rle()x

rle()"rle"
lengths
values      lengths
inverse.rle()

x <- rev(rep(6:10, 1:5))
rle(x)
## lengths [1:5]  5 4 3 2 1
## values  [1:5] 10 9 8 7 6

z <- c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE)
rle(z)
rle(as.character(z))
print(rle(z), prefix = "..| ")

N <- integer(0)
stopifnot(x == inverse.rle(rle(x)),
          identical(N, inverse.rle(rle(N))),
          z == inverse.rle(rle(z)))

```

Round

```

ceilingxx
floorxx
truncxx0
round
signifnumericxsignif(x, dig)round(x, dig - ceiling(log10(abs(x))))

ceiling(x)
floor(x)
trunc(x, ...)

round(x, digits = 0, ...)
signif(x, digits = 6)

```

```

x          roundsignif
digits     roundsignifround
...

```

Math

```

round(0.5)0round(-1.5)-20.15round(0.15, 1)0.10.2
round(x, digits = -2)
signifdigits1...22

```

```

ceilingfloortruncMathtrunc
roundsignifMath2

```

```

floorceilingfloor(log(x, base = 8))x = 810
digits != 0round(x, d)d > 0sprintf()format()

```

<https://www.iso.org>
https://en.wikipedia.org/wiki/IEEE_754

```

as.integerroundX()version = "3d.C"

```

```

round(.5 + -2:4) # IEEE / IEC rounding: -2 0 0 2 2 4 4
## (this is *good* behaviour -- do *NOT* report it as bug !)

```

```

( x1 <- seq(-2, 4, by = .5) )
round(x1) #-- IEEE / IEC rounding !
x1[trunc(x1) != floor(x1)]
x1[round(x1) != floor(x1 + .5)]
(non.int <- ceiling(x1) != floor(x1))

```

```

x2 <- pi * 100^(-1:3)
round(x2, 3)
signif(x2, 3)

```

round.POSIXt

```
## S3 method for class 'POSIXt'
round(x,
      units = c("secs", "mins", "hours", "days", "months", "years"))
## S3 method for class 'POSIXt'
trunc(x,
      units = c("secs", "mins", "hours", "days", "months", "years"),
      ...)

## S3 method for class 'Date'
round(x, ...)
## S3 method for class 'Date'
trunc(x,
      units = c("secs", "mins", "hours", "days", "months", "years"),
      ...)
```

```
x          "POSIXt""Date"
units
...        digitsround
```

```
units"Date"
```

```
"POSIXlt""Date"
```

```
round
```

```
DateTimeClassesDate
```

```
round(.leap.seconds + 1000, "hour")
```

```
      trunc(Sys.time(), "day")
(timM <- trunc(Sys.time() -> St, "months")) # shows timezone
(datM <- trunc(Sys.Date() -> Sd, "months"))
(timY <- trunc(St, "years")) # + timezone
(datY <- trunc(Sd, "years"))
```

```
stopifnot(inherits(datM, "Date"), inherits(timM, "POSIXt"),
          substring(format(datM), 9,10) == "01", # first of month
          substring(format(datY), 6,10) == "01-01", # Jan 1
          identical(format(datM), format(timM)),
          identical(format(datY), format(timY)))
```

row

```
row(x, as.factor = FALSE)
.row(dim)
```

```
x          dim
dim
as.factor
```

```
xijii
```

[colslice.index](#)

```
x <- matrix(1:12, 3, 4)
# extract the diagonal of a matrix - more slowly than diag(x)
dx <- x[row(x) == col(x)]
dx

# create an identity 5-by-5 matrix more slowly than diag(n = 5):
x <- matrix(0, nrow = 5, ncol = 5)
x[row(x) == col(x)] <- 1
x

(i34 <- .row(3:4))
stopifnot(identical(i34, .row(c(3,4)))) # 'dim' maybe "double"
```

row+colnames

```
rownames(x, do.NULL = TRUE, prefix = "row")
rownames(x) <- value
```

```
colnames(x, do.NULL = TRUE, prefix = "col")
colnames(x) <- value
```

```

x          colnames
do.NULL    FALSENULL
prefix
value      dimnames(x)NULL

```

```

xdimnamesrownamescolnamesrow.namesnames
do.NULLFALSENROW(x)NCOL(x)prefixNULL
valueas.character
valuerownamescolnamesvalueas.charactercolnames

```

```

      rownames(x)[3] <- "c"

xvalueNULLrownames(x)

```

```

dimnamescase.namesvariable.names

```

```

m0 <- matrix(NA, 4, 0)
rownames(m0)

m2 <- cbind(1, 1:4)
colnames(m2, do.NULL = FALSE)
colnames(m2) <- c("x","Y")
rownames(m2) <- rownames(m2, do.NULL = FALSE, prefix = "Obs.")
m2

```

```

row.names

```

```

data.frame
`.rowNamesDF<-`make.namesrow.names<-

```

```

row.names(x)
row.names(x) <- value
.rowNamesDF(x, make.names=FALSE) <- value

```

```

x          "data.frame"
make.names logicalFALSE,    NA,    TRUEvalueNAFALSENATRUEmake.names(value,
unique=TRUE)
value      xNULL

```

```

row.namesattr(x, "row.names")
NULLseq_len(nrow(x))

row.names
row.names<-

row.namesrownamesrownames
1:nn > 2row.namesattr(x, "row.names")as.matrixdata.matrix

```

```

data.framerownamesnames
.row_names_info

```

```

## To illustrate the note:
df <- data.frame(x = c(TRUE, FALSE, NA, NA), y = c(12, 34, 56, 78))
row.names(df) <- 1 : 4
attr(df, "row.names") #> 1:4
deparse(df) # or dput(df)
##--> c(NA, 4L) : Compact storage, *not* regarded as automatic.

row.names(df) <- NULL
attr(df, "row.names") #> 1:4
deparse(df) # or dput(df) -- shows
##--> c(NA, -4L) : Compact storage, regarded as automatic.

```

```
rowsum
```

```
rowsum
```

```

rowsum(x, group, reorder = TRUE, ...)

## S3 method for class 'data.frame'
rowsum(x, group, reorder = TRUE, na.rm = FALSE, ...)

## Default S3 method:
rowsum(x, group, reorder = TRUE, na.rm = FALSE, ...)

```

```

x
group          x
reorder        TRUEsort(unique(group))FALSE
na.rm          TRUEFALSENaN
...

tapplygroupx

groupcolSums
NA

group

tapplyaggaterowSums

require(stats)

x <- matrix(runif(100), ncol = 5)
group <- sample(1:8, 20, TRUE)
(xsum <- rowsum(x, group))
## Slower versions
tapply(x, list(group[row(x)], col(x)), sum)
t(sapply(split(as.data.frame(x), group), colSums))
aggregate(x, list(group), sum)[-1]

```

S3method

```
.S3method(generic, class, method)
```

```

generic
class
method

```

```
S3methodNAMESPACE
```

```
## Create a generic function and register a method for objects
## inheriting from class 'cls':
gen <- function(x) UseMethod("gen")
met <- function(x) writeLines("Hello world.")
.S3method("gen", "cls", met)
## Create an object inheriting from class 'cls', and call the
## generic on it:
x <- structure(123, class = "cls")
gen(x)
```

sample

samplex

```
sample(x, size, replace = FALSE, prob = NULL)
```

```
sample.int(n, size = n, replace = FALSE, prob = NULL,
           useHash = (n > 1e+07 && !replace && is.null(prob) && size <= n/2))
```

x

n

size

replace

prob

useHash **logical** replace = FALSE prob = NULL size <= n/2 nuseHash=FALSE n

```
is.numeric x >= 1 sample1:xx sample(x)
```

xlength

```
samplesize sample(x) x1:x
```

```
size = 0 n = 0 x n > 0 length(x)
```

```
nx.Machine$integer.max
```

prob replace

replace size

sample.int n size

n integer double

samplesize x1:x

sample.int size1:n $n \geq 2^{31}$


```
RNGkind(sample.kind = ..)sample()
```

```
x <- 1:12
# a random permutation
sample(x)
# bootstrap resampling -- only if length(x) > 1 !
sample(x, replace = TRUE)

# 100 Bernoulli trials
sample(c(0,1), 100, replace = TRUE)

## More careful bootstrapping -- Consider this when using sample()
## programmatically (i.e., in your function or simulation)!

# sample()'s surprise -- example
x <- 1:10
  sample(x[x > 8]) # length 2
  sample(x[x > 9]) # oops -- length 10!
  sample(x[x > 10]) # length 0

## safer version:
resample <- function(x, ...) x[sample.int(length(x), ...)]
resample(x[x > 8]) # length 2
resample(x[x > 9]) # length 1
resample(x[x > 10]) # length 0

## R 3.0.0 and later
sample.int(1e10, 12, replace = TRUE)
sample.int(1e10, 12) # not that there is much chance of duplicates
```

save

```
saveloadattachdata
```

```
save.image()save(list = ls(all.names = TRUE), file = ".RData", envir = .GlobalEnv)
q("yes")
```

```
save(..., list = character(),
  file = stop("'file' must be specified"),
  ascii = FALSE, version = NULL, envir = parent.frame(),
  compress = isTRUE(!ascii), compression_level,
  eval.promises = TRUE, precheck = TRUE)
```

```
save.image(file = ".RData", version = NULL, ascii = FALSE,
  compress = !ascii, safe = TRUE)
```

```

...
list          NULL
file          save.imageversion = 1
ascii         TRUEasciiFALSENAversion >= 2
version       NULL
envir
compress      TRUEgzip"gzip""bzip2""xz""zstd"file
compression_level
              6gzip9bzip2xzfile
eval.promises
precheck
safe          TRUEfilefile

...listenvireval.promises = FALSE

asciicompresssafeversion"save.defaults"savesave.image"save.image.defaults"
"save.defaults"save.imagecompression_level"save.defaults"
"wb"

gzipbzip2xzxz
gzipbzip2xzcompress = FALSEresaveRdaFiles

filepigzhttps://zlib.net/pigz/pbzip2https://launchpad.net/pbzip2pipe

con <- pipe("pigz -p8 > fname.gz", "wb")
save(myObj, file = con); close(con)

con <- pipe("pbzip2 -p8 -9 > fname.bz2", "wb")
save(myObj, file = con); close(con)

con <- pipe("xz -T8 -6 -e > fname.xz", "wb")
save(myObj, file = con); close(con)

xz

...envir
ascii = TRUE

ASCII = NAscanf
compress = "zstd"

```

```
saveRDS()save()readRDS()load()
```

```
save.image
```

```
Error in gzfile(file, "wb") : unable to open connection
In addition: Warning message:
In gzfile(file, "wb") :
  cannot open compressed file '.RDataTmp',
  probable reason 'Permission denied'
```

```
dputdumploaddata
```

```
serializesaveRDS
```

```
x <- stats::runif(20)
y <- list(a = 1, b = TRUE, c = "oops")
save(x, y, file = "xy.RData")
save.image() # creating ".RData" in current working directory
unlink("xy.RData")

# set save defaults using option:
options(save.defaults = list(ascii = TRUE, safe = FALSE))
save.image() # creating ".RData"
if(interactive()) withAutoprint({
  file.info(".RData")
  readLines(".RData", n = 7) # first 7 lines; first starts w/ "RDA"..
})
unlink(".RData")
```

```
scale
```

```
scale
```

```
scale(x, center = TRUE, scale = TRUE)
```

```
x
center      xas.numeric(.)is.numeric(.)
scale       x
```

```
centercenterxxcentercenterTRUENAxcenterFALSE
```

```
scalescalexxscalescaleTRUExcenterTRUEscaleFALSE
```

```
 $\sqrt{\sum(x^2)/(n-1)}$ xncenter = TRUEscale(x, center = FALSE, scale = apply(x, 2, sd,
na.rm = TRUE))
```

```
scale.default"scaled:center""scaled:scale"
```

sweep

par

```
require(stats)
x <- matrix(1:10, ncol = 2)
(centered.x <- scale(x, scale = FALSE))
cov(centered.scaled.x <- scale(x)) # all 1
```

scan

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
      quote = if(identical(sep, "\n")) "" else "'\\'", dec = ".",
      skip = 0, nlines = 0, na.strings = "NA",
      flush = FALSE, fill = FALSE, strip.white = FALSE,
      quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,
      comment.char = "", allowEscapes = FALSE,
      fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

file	"" stdin()Ctrl-DCtrl-Z getwd()file = "stdin"stdin file fileconnectionsep = "\n" fileurl fileencodingscanfileEncoding
what	whattypeoflogicalintegernumericcomplexcharacterrawlistwhat length(what)NULL
nmax	whatnlinesscanfile
n	
sep	sep NULL
quote	NULL
dec	NULL
skip	

```

nlines
na.strings      NA
na.strings      NA
flush           TRUE
flush           TRUE
fill            TRUE
fill            TRUE
strip.white     what
strip.white     what
strip.white     strip.white[i]TRUEiwhat[i]i
quiet           FALSE
blank.lines.skip TRUE
blank.lines.skip TRUE
multi.line      what
multi.line      what
comment.char    ""
comment.char    ""
allowEscapes    \n
allowEscapes    \a, \b, \f, \n, \r, \t, \v\040\0x2A\u\U
fileEncoding    file
fileEncoding    file
encoding        "latin1""UTF-8"fileEncoding
text            file
text            file
skipNul

```

```

whatscanwhatNULLNULL
whatNULLis.atomic
"\n""\xa0"
na.strings""
NANaInfinfinityInf-Inf0
NA0-9NA_integer_
sep""\
sep.csv'""sep = "\n"
NULL
sepscan("foo", sep = "\n", blank.lines.skip = FALSE)"\n"readLines
comment.char

```

```

what
filefileEncodingencodingencodingscanreadLines
scanskipNul = TRUE

```

```

whatwhat
what
encoding"latin1""UTF-8"

```

```
multi.lineflush = TRUEmulti.line = FALSE
nnmaxnmaxnlines
scan
nul\0allowEscapes = TRUEnulreadBin
```

[read.tablereadLineswrite](#)

Quotes

[readCharreadBin](#)

```
cat("TITLE extra line", "2 3 5 7", "11 13 17", file = "ex.data", sep = "\n")
pp <- scan("ex.data", skip = 1, quiet = TRUE)
scan("ex.data", skip = 1)
scan("ex.data", skip = 1, nlines = 1) # only 1 line after the skipped one
scan("ex.data", what = list("", "", "")) # flush is F -> read "7"
scan("ex.data", what = list("", "", ""), flush = TRUE)
unlink("ex.data") # tidy up

## "inline" usage
scan(text = "1 2 3")
```

search

[attachlibrarydata.frames](#)

```
search()
searchpaths()
```

```
".GlobalEnv""package:base"
searchpaths
```

```
search
searchpaths
```

[.packagespath.package](#)
[loadedNamespaces](#)
[attachdetachobjects](#)

```
search()
searchpaths()
```

seek

```
seek(con, ...)  
## S3 method for class 'connection'  
seek(con, where = NA, origin = "start", rw = "", ...)
```

```
isSeekable(con)
```

```
truncate(con, ...)
```

```
con  
where          originNA  
rw             "read""write"  
origin         "start""current""end"  
...
```

```
seekwhere = NAwhereisSeekableseek  
whereinteger  
seekrw  
"r+" "r+b"  
gzfileseekorigin = "end"  
seekNAwhere  
truncatefile
```

```
seek0integer  
truncateNULL  
isSeekableseek
```

seek

[connections](#)

seq

seqseq.intseq_alongseq_len

seq(...)

Default S3 method:

```
seq(from = 1, to = 1, by = ((to - from)/(length.out - 1)),  
    length.out = NULL, along.with = NULL, ...)
```

seq.int(from, to, by, length.out, along.with, ...)

seq_along(along.with)
seq_len(length.out)

...

fromto 1from

by

length.out seqseq.int

along.with

[NaN](#)NA

seqseq.int

seqalong.withseq_along

seq.int"seq"

seq(from, to)

seq(from, to, by=)

seq(from, to, length.out=)

seq(along.with=)

seq(from)

seq(length.out=)

from, from+/-1, ..., tofrom:to

from, from+bytoto - frombytoto10⁻¹⁰abs(from - to)

length.outfromtolength.outlengthlenseq_len

1, 2, ..., length(along.with)along.withalongseq_along

1, 2, ..., length(from)along.with1:fromseq(0)seq_alongseq_len

1, 2, ..., length.outlength.out = 0integer(0)

from - to10⁻¹⁴from

seqfromtobylength.outalong.withseqMathOpsSummary

seq.intseq_alongseq_len


```
seq.intseq"integer""double"  
seq_alongseq_len
```

```
seq.Dateseq.POSIXt  
:repsequencerowcol
```

```
seq(0, 1, length.out = 11)  
seq(stats::rnorm(20)) # effectively 'along'  
seq(1, 9, by = 2)      # matches 'end'  
seq(1, 9, by = pi)     # stays below 'end'  
seq(1, 6, by = 3)  
seq(1.575, 5.125, by = 0.05)  
seq(17) # same as 1:17, or even better seq_len(17)
```

seq.Date

```
seq"Date"
```

```
## S3 method for class 'Date'  
seq(from, to, by, length.out = NULL, along.with = NULL, ...)
```

```
from  
to  
by  
length.out  
along.with  
...
```

```
by
```

```
difftime  
"day""week""month""quarter""year"pmatch()"s"  
seq.POSIXt"month"  
seq(from, to)by"day""1 day"
```

"Date""integer"object.size

Date

```
## first days of years
seq(as.Date("1910/1/1"), as.Date("1999/1/1"), "years")
## by month
seq(as.Date("2000/1/1"), by = "month", length.out = 12)
## quarters
seq(as.Date("2000/1/1"), as.Date("2003/1/1"), by = "quarter")

## 3-week period ending on a fixed date
seq(to = as.Date("2024-06-18"), by = "day", length.out = 21)

## find all 7th of the month _strictly_ inside two dates, the last being a 7th.
st <- as.Date("1998-12-17")
en <- as.Date("2000-1-7")
ll <- seq(en, st, by = "-1 month")
rev(ll[st < ll & ll < en])

## can abbreviate 'month' to 'm':
identical(seq(st, en, by = "m"),
          seq(st, en, by = "1 month"))
```

seq.POSIXt

seq"POSIXt"

```
## S3 method for class 'POSIXt'
seq(from, to, by, length.out = NULL, along.with = NULL, ...)

from
to
by
length.out
along.with
...

by

difftime
"sec""min""hour""day""DSTday""week""month""quarter""year"pmatch()"s"
"day""DSTday""week""7 DSTdays""month""year"
fromto
"month"
```

"POSIXct""double""integer"

DateTimeClasses

```
## first days of years
seq(ISOdate(1910,1,1), ISOdate(1999,1,1), "years")
## by month
seq(ISOdate(2000,1,1), by = "month", length.out = 12)
seq(ISOdate(2000,1,31), by = "month", length.out = 4)
## quarters
seq(ISOdate(1990,1,1), ISOdate(2000,1,1), by = "quarter") # or "3 months"
## days vs DSTdays: use c() to lose the time zone.
seq(c(ISOdate(2000,3,20)), by = "day", length.out = 10)
seq(c(ISOdate(2000,3,20)), by = "DSTday", length.out = 10)
seq(c(ISOdate(2000,3,20)), by = "7 DSTdays", length.out = 4)

## 24-hour period ending at a fixed time
seq(to = ISOdate(2024,1,2, 3,4,5), by = "hour", length.out = 24)
```

sequence

sequence**seq**(from[i], by = by[i], length.out = nvec[i])ifrombynvec

```
sequence(nvec, ...)
## Default S3 method:
sequence(nvec, from = 1L, by = 1L, ...)
```

nvec
from
by
...

frombysequence(nvec, from, by=0L)rep(from, each=nvec)

glseqrep

```

sequence(c(3, 2)) # the concatenated sequences 1:3 and 1:2.
#> [1] 1 2 3 1 2
sequence(c(3, 2), from=2L)
#> [1] 2 3 4 2 3
sequence(c(3, 2), from=2L, by=2L)
#> [1] 2 4 6 2 4
sequence(c(3, 2), by=c(-1L, 1L))
#> [1] 1 0 -1 1 2

```

serialize

```

serialize(object, connection, ascii, xdr = TRUE,
          version = NULL, refhook = NULL)

```

```

unserialize(connection, refhook = NULL)

```

```

object
connection      serializeNULLunserialize
ascii           TRUEENASave
xdr
version         NULL
refhook

```

```

serializeobjectconnectionNULLobjectserialize
serialize
unserializeserializeconnection
refhook.GlobalEnvserializeNULLunserializeserialize
asciiTRUEascii = FALSE
XAreadRDS
xdr = FALSE

```

```

serializeNULLconnection = NULL
unserialize

```

[saveRDS](#)
[saveload](#)

```
x <- serialize(list(1,2,3), NULL)
unserialize(x)

## see also the examples for saveRDS
```

sets

```
union(x, y)
intersect(x, y)
setdiff(x, y)
setequal(x, y)

is.element(el, set)
```

xyelset

```
is.vector()
as.vector()isa(x, class(y))isa(y, class(x))dim(x)xy0L
is.element(x, y)x %in% yas.vector()
```

```
union
intersectNULLxyNULL
setdiffmodex
setequalxis.element
```

[%in%](#)
unionintersect

```

(x <- c(sort(sample(1:20, 9)), NA))
(y <- c(sort(sample(3:23, 7)), NA))
union(x, y)
intersect(x, y)
setdiff(x, y)
setdiff(y, x)
setequal(x, y)

## True for all possible x & y :
setequal( union(x, y),
          c(setdiff(x, y), intersect(x, y), setdiff(y, x)))

is.element(x, y) # length 10
is.element(y, x) # length 8

## Factors:
x <- as.factor(c("A", "B", "A"))
y <- as.factor(c("B", "b"))
union(x, y)
intersect(x, y)
setdiff(x, y)
setdiff(y, x)
## (Note that union() and intersect() merge the levels.)

```

setTimeLimit

```
setTimeLimit(cpu = Inf, elapsed = Inf, transient = FALSE)
```

```
setSessionTimeLimit(cpu = Inf, elapsed = Inf)
```

```
cpuelapsed
```

```
transient      TRUE
```

```
setTimeLimittransient = TRUE
```

```
setSessionTimeLimitInf
```

[Sys.sleep](#)

showConnections

```
showConnections(all = FALSE)
getConnection(what)
closeAllConnections()
```

```
stdin()
stdout()
stderr()
nullfile()
```

```
isatty(con)
```

```
getAllConnections()
```

```
all
```

```
what          showConnections
```

```
con
```

```
stdin()stdout()stderr()"terminal"stdout()stderr()sinkstdout()
stdin()--encoding
nullfile()"/dev/null""nul:"
showConnectionsgetConnectiongzcon
closeAllConnectionssink
isatty"terminal"
getAllConnectionsgetConnectionshowConnections(all = TRUE)
```

```
stdin()stdout()stderr()
showConnections
getConnectionNULL
```

```
stdin()stdinstdinstdinfile("stdin")
```

[connections](#)

```

showConnections(all = TRUE)
## Not run:
textConnection(letters)
# oops, I forgot to record that one
showConnections()
# class      description      mode text  isopen  can read can write
#3 "letters" "textConnection" "r"  "text" "opened" "yes"    "no"
mycon <- getConnection(3)

## End(Not run)

c(isatty(stdin()), isatty(stdout()), isatty(stderr()))

```

shQuote

```
shQuote(string, type = c("sh", "csh", "cmd", "cmd2"))
```

```

string
type      "cmd""cmd2""cmd"

```

```
shbashkshzsh
```

```
cshtcsh
```

```

https://learn.microsoft.com/en-us/cpp/c-language/parsing-c-command-line-arguments?
view=msvc-160type = "cmd"shQuotesystemsystem2

```

```
cmd.exeshelltype = "cmd2""^"type = "cmd2"cmd.exe
```

```
string
```

[sQuote](#)


```

test <- "abc$def`gh`i\\j"
cat(shQuote(test), "\n")
## Not run: system(paste("echo", shQuote(test)))
test <- "don't do it!"
cat(shQuote(test), "\n")

tryit <- paste("use the", sQuote("-c"), "switch\nlike this")
cat(shQuote(tryit), "\n")
## Not run: system(paste("echo", shQuote(tryit)))
cat(shQuote(tryit, type = "csh"), "\n")

## Windows-only example, assuming cmd.exe:
perlcmd <- 'print "Hello World\\n";'
## Not run:
shell(shQuote(paste("perl -e",
                    shQuote(perlcmd, type = "cmd")),
      type = "cmd2"))

## End(Not run)

```

sign

signx-1

sign

sign(x)

x

Math

abs

```

sign(pi)      # == 1
sign(-2:3)    # -1 -1 0 1 1 1

```

Signals

SIGUSR1SIGUSR2.[Laston.exit](#)

```
kill -USR1 pid
kill -USR2 pid
```

pid

[Sys.getpid](#)

sink

```
sink
sink.number()
sink.number(type = "message")
```

```
sink(file = NULL, append = FALSE, type = c("output", "message"),
      split = FALSE)
```

```
sink.number(type = c("output", "message"))
```

file	NULL
append	TRUEfilefile
type	
split	TRUEtee

```
sinkfile
stdouttype = "output"stderr()messagewarningstopsink(type = "message")
sink()sink(file = NULL)
file"wt"
split = TRUEvprintfwritelinesstdout()
file
filefile
```

```
sinkNULL
sink.number()
sink.number("message")
```

```
sink
```

```
capture.output
```

```
sink("sink-examp.txt")
i <- 1:10
outer(i, i)
sink()
```

```
## capture all the output to a file.
zz <- file("all.Rout", open = "wt")
sink(zz)
sink(zz, type = "message")
try(log("a"))
## revert output back to the console -- only then access the file!
sink(type = "message")
sink()
file.show("all.Rout", delete.file = TRUE)
```

slice.index

slice.index(x, MARGIN)

x x
MARGIN

$$i_1 + \dots + n_k(i_k - 1) + 1$$

yx

rowcolslice.indexMARGINx

```
x <- array(1 : 24, c(2, 3, 4))
slice.index(x, 2)
slice.index(x, c(1, 3))
## When slicing by dimensions 1 and 3, slice index 5 is obtained for
## dimension 1 has value 1 and dimension 3 has value 3 (see above):
which(slice.index(x, c(1, 3)) == 5, arr.ind = TRUE)
```

slotOp

object@name
object@name <- value

object
name objectnameobject
value object

```
object
object.Data$slot@
```

Extractslot

socketSelect

```
socketSelect(socklist, write = FALSE, timeout = NULL)
```

```
socklist
write      TRUE
timeout    NULLNULL
```

```
writesocklistsocklist
```

```
socklistwrite
```

```
## Not run:
## test whether socket connection s is available for writing or reading
socketSelect(list(s, s), c(TRUE, FALSE), timeout = 0)

## End(Not run)
```

solve

`a %*% x = bxb`

`solve(a, b, ...)`

Default S3 method:

`solve(a, b, tol, LINPACK = FALSE, ...)`

`a`

`b` `bsolvea`

`tol` `a.Machine$double.eps`

`LINPACK`

`...`

`ab`

`abbaab`

`aqr.solveqr.solve`

`abNaN`

`toltol <= 0`

`a"qr"solve.qr`

DGESVZGESV

<https://netlib.org/lapack/>

https://netlib.org/lapack/lug/lapack_lug.html

`solve.qrqrchol2invbacksolveqr.solve`

`hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }`

`h8 <- hilbert(8); h8`

`sh8 <- solve(h8)`

`round(sh8 %*% h8, 3)`

`A <- hilbert(4)`

`A[] <- as.complex(A)`

might not be supported on all platforms

`try(solve(A))`

sort

[order](#)

```
sort(x, decreasing = FALSE, ...)
```

```
## Default S3 method:
```

```
sort(x, decreasing = FALSE, na.last = NA, ...)
```

```
sort.int(x, partial = NULL, na.last = NA, decreasing = FALSE,  
         method = c("auto", "shell", "quick", "radix"), index.return = FALSE)
```

x	sortsort.int
decreasing	
...	sort.int
na.last	NATRUEFALSENA
partial	NULL
method	
index.return	method == "radix"na.lastna.last = NA

```
sortsort.int  
sortorderxtfrmxtfrmis.numeric(x)
```

```
"auto""radix"231"shell"
```

```
"radix"Comparison
```

```
partialNULLpartial
```

```
"shell" $O(n^{4/3})$ x
```

```
"quick"xpartialNULLx
```

```
"radix"
```

```
"radix"na.last
```

```
xcharacterLC_COLLATE=C
```

```
231complex
```

```
sortxsort.intx[order(x, ...)]\[orderxtfrm
```

```
sort.intindex.returnxixmethod == "quick"sort.listmethod == "radix"index.return
```

```
na.lastindex.returnna.lastNANAorder
```

```
partial
```

order

is.unsortedrank

```
require(stats)

x <- swiss$Education[1:25]
x; sort(x); sort(x, partial = c(10, 15))

## illustrate 'stable' sorting (of ties):
sort(c(10:3, 2:12), method = "shell", index.return = TRUE) # is stable
## $x : 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 12
## $ix: 9 8 10 7 11 6 12 5 13 4 14 3 15 2 16 1 17 18 19
sort(c(10:3, 2:12), method = "quick", index.return = TRUE) # is not
## $x : 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 12
## $ix: 9 10 8 7 11 6 12 5 13 4 14 3 15 16 2 17 1 18 19

x <- c(1:3, 3:5, 10)
is.unsorted(x) # FALSE: is sorted
is.unsorted(x, strictly = TRUE) # TRUE : is not (and cannot be)
# sorted strictly

## Not run:
## Small speed comparison simulation:
N <- 2000
Sim <- 20
rep <- 1000 # << adjust to your CPU
c1 <- c2 <- numeric(Sim)
for(is in seq_len(Sim)){
  x <- rnorm(N)
  c1[is] <- system.time(for(i in 1:rep) sort(x, method = "shell"))[1]
  c2[is] <- system.time(for(i in 1:rep) sort(x, method = "quick"))[1]
  stopifnot(sort(x, method = "shell") == sort(x, method = "quick"))
}
rbind(ShellSort = c1, QuickSort = c2)
cat("Speedup factor of quick sort():\n")
summary({qq <- c1 / c2; qq[is.finite(qq)]})

## A larger test
x <- rnorm(1e7)
system.time(x1 <- sort(x, method = "shell"))
system.time(x2 <- sort(x, method = "quick"))
system.time(x3 <- sort(x, method = "radix"))
stopifnot(identical(x1, x2))
stopifnot(identical(x1, x3))

## End(Not run)
```

sort_by

```
sort_by(x, y, ...)  
  
## Default S3 method:  
sort_by(x, y, ...)  
  
## S3 method for class 'data.frame'  
sort_by(x, y, ...)  
  
x  
y  
  xtrm  
data.frame ~ g~ list(g)g ~ g1 + ... + gk~ list(g1, ..., gk)g1gkx  
y = gy = ~ gy = list(g1, ..., gk)y = ~ list(g1, ..., gk)x  
...  orderorder  
  
xxxy  
  
sortorder  
  
mtcars$am  
mtcars$mpg  
with(mtcars, sort_by(mpg, am)) # group mpg by am  
  
## data.frame method  
sort_by(mtcars, runif(nrow(mtcars))) # random row permutation  
sort_by(mtcars, list(mtcars$am, mtcars$mpg))  
  
# formula interface  
sort_by(mtcars, ~ am + mpg) |> subset(select = c(am, mpg))  
sort_by.data.frame(mtcars, ~ list(am, -mpg)) |> subset(select = c(am, mpg))
```

source

source[parse](#)

```
withAutoprint(exprs)source(exprs = exprs, ..)
```

```
source(file, local = FALSE, echo = verbose, print.eval = echo,
      exprs, spaced = use_file,
      verbose = getOption("verbose"),
      prompt.echo = getOption("prompt"),
      max.deparse.length = 150, width.cutoff = 60L,
      deparseCtrl = "showAttributes",
      chdir = FALSE,
      catch.aborts = FALSE,
      encoding = getOption("encoding"),
      continue.echo = getOption("continue"),
      skip.echo = 0, keep.source = getOption("keep.source"))
```

```
withAutoprint(exprs, evaluated = FALSE, local = parent.frame(),
      print. = TRUE, echo = TRUE, max.deparse.length = Inf,
      width.cutoff = max(20, getOption("width")),
      deparseCtrl = c("keepInteger", "showAttributes", "keepNA"),
      skip.echo = 0,
      ...)
```

file [stdin\(\)](#)

local TRUEFALSEFALSETRUEsource

echo TRUE

print.evalprint.

TRUEEval(i)iecho

exprs source()withAutoprint(*, evaluated=TRUE)file[expressioncalllist](#)
[call](#)

withAutoprint()evaluated=FALSE

evaluated exprssource(exprs= *)expressioncalllist

spaced echo = TRUE

verbose TRUEEcho = TRUE

prompt.echo echo = TRUE

max.deparse.length

echoTRUE

width.cutoff [deparse\(\)](#)

deparseCtrl [character](#)control[deparse\(\).deparseOpts](#)"showAttributes"

deparseCtrl = "all"

chdir TRUEfilefile

catch.aborts

```

encoding      filefile"unknown"
continue.echo echo = TRUE
skip.echo     echo = TRUE
keep.source
...           withAutoprint()source(.)

```

```

sourceprinttraceback()withVisible

```

```

keep.source

```

```

skip.echo > 0skip.echo
echomax.deparse.length .... [TRUNCATED]

```

```

fileencoding = "unknown"localeToCharset()encodingencoding"unknown"Encoding
filesourceneencodingparse

```

```

demosourceevalparsescanoptions("keep.source")
sys.source

```

```

someCond <- 7 > 6
## want an if-clause to behave "as top level" wrt auto-printing :
## (all should look "as if on top level", e.g. non-assignments should print:)
if(someCond) withAutoprint({
  x <- 1:12
  x-1
  (y <- (x-5)^2)
  z <- y
  z - 10
})

```

```

## If you want to source() a bunch of files, something like
## the following may be useful:
sourceDir <- function(path, trace = TRUE, ...) {
  op <- options(); on.exit(options(op)) # to reset after each
  for (nm in list.files(path, pattern = "[.]?[RrSsQq]$", ...)) {
    if(trace) cat(nm,":")
    source(file.path(path, nm), ...)
    if(trace) cat("\n")
    options(op)
  }
}

```

```

}

suppressWarnings( rm(x,y) ) # remove 'x' or 'y' from global env
withAutoprint({ x <- 1:2; cat("x=",x, "\n"); y <- x^2 })
## x and y now exist:
stopifnot(identical(x, 1:2), identical(y, x^2))

withAutoprint({ formals(sourceDir); body(sourceDir) },
              max.deparse.length = 20, verbose = TRUE)

## Continuing after (catchable) errors:
tc <- textConnection('1:3
2 + "3"
cat(" .. in spite of error: happily continuing! ..\n")
6*7')
r <- source(tc, catch.aborts = TRUE)
## Error in 2 + "3" ....
## .. in spite of error: happily continuing! ..
stopifnot(identical(r, list(value = 42, visible=TRUE)))

```

Special

```

beta(a, b)
lbeta(a, b)

gamma(x)
lgamma(x)
psigamma(x, deriv = 0)
digamma(x)
trigamma(x)

choose(n, k)
lchoose(n, k)
factorial(x)
lfactorial(x)

ab
xn
kderiv

betalbeta

```

$$B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

$$B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt$$

$$\text{gamma}\backslash\text{gamma}\Gamma(x)$$

$$x > 0 \Gamma(x+1) = x\Gamma(x)\Gamma(x+n) = (x+n-1)(x+n-2)\cdots x\Gamma(x)n\Gamma(x)$$

NaN10⁻⁸-10

$$\text{factorial}(x)x!\text{xgamma}(x+1)\text{lfactorial}\text{lgamma}(x+1)$$
$$\text{digammatrixgamma}(x, \text{deriv}) \text{deriv} \geq 0 \text{deriv} \psi(x)$$

```
psi_gamma_deriv = 2:4
```

$$n! = n(n-1)(n-2)\cdots(n-k+1)(n-k+2)\cdots 1$$

```
choose(*, k)[1]gammakcombn
```

gammalgammadigammatrigammaMath

gammalgammabetalbeta

```
digammatrigammapsigmax >= 0
```

$$x < 0 \text{deriv} \leq 5$$

gamma1gamma

https://en.wikipedia.org/wiki/Abramowitz_and_Stegun

ArithmeticsqrtBessel

pgamma

```
require(graphics)
```

```
choose(5, 2)
```

```
for (n in 0:10) print(choose(n, k = 0:n))
```

```
factorial(100)
```

```
lfactorial(10000)
```

```
## gamma has 1st order poles at 0, -1, -2, ...
```

```
## this will generate loss of precision warnings, so turn off
```

```
op <- options("warn")
```

```
options(warn = -1)
```

```
x <- sort(c(seq(-3, 4, length.out = 201), outer(0:-3, (-1:1)*1e-6, `+`)))
```

```

plot(x, gamma(x), ylim = c(-20,20), col = "red", type = "l", lwd = 2,
     main = expression(Gamma(x)))
abline(h = 0, v = -3:0, lty = 3, col = "midnightblue")
options(op)

x <- seq(0.1, 4, length.out = 201); dx <- diff(x)[1]
par(mfrow = c(2, 3))
for (ch in c("", "l", "di", "tri", "tetra", "penta")) {
  is.deriv <- nchar(ch) >= 2
  nm <- paste0(ch, "gamma")
  if (is.deriv) {
    dy <- diff(y) / dx # finite difference
    der <- which(ch == c("di", "tri", "tetra", "penta")) - 1
    nm2 <- paste0("psigamma(*, deriv = ", der, ")")
    nm <- if(der >= 2) nm2 else paste(nm, nm2, sep = " ==\n")
    y <- psigamma(x, deriv = der)
  } else {
    y <- get(nm)(x)
  }
  plot(x, y, type = "l", main = nm, col = "red")
  abline(h = 0, col = "lightgray")
  if (is.deriv) lines(x[-1], dy, col = "blue", lty = 2)
}
par(mfrow = c(1, 1))

## "Extended" Pascal triangle:
fN <- function(n) formatC(n, width=2)
for (n in -4:10) {
  cat(fN(n), ":", fN(choose(n, k = -2:max(3, n+2))))
  cat("\n")
}

## R code version of choose() [simplistic; warning for k < 0]:
mychoose <- function(r, k)
  ifelse(k <= 0, (k == 0),
         sapply(k, function(k) prod(r:(r-k+1))) / factorial(k))
k <- -1:6
cbind(k = k, choose(1/2, k), mychoose(1/2, k))

## Binomial theorem for n = 1/2 ;
## sqrt(1+x) = (1+x)^(1/2) = sum_{k=0}^Inf choose(1/2, k) * x^k :
k <- 0:10 # 10 is sufficient for ~ 9 digit precision:
sqrt(1.25)
sum(choose(1/2, k) * .25^k)

```

split

splitxfunsplitsplit

split(x, f, drop = FALSE, ...)

```

## Default S3 method:
split(x, f, drop = FALSE, sep = ".", lex.order = FALSE, ...)

split(x, f, drop = FALSE, ...) <- value
unsplit(value, f, drop = FALSE)

x
f          as.factor(f)xf ~ gg ~ g1 + ... + gkg1gkx
drop       ffactor
value      x
sep        interactionlist
lex.order  interactionf
...

splitsplit<-data.frame
unsplitsplitfvalue
fxf
fx
interactionlist.seplevels

splitfdrop = TRUE
unsplitsplit(x, f)value

cut
strsplit

require(stats); require(graphics)
n <- 10; nn <- 100
g <- factor(round(n * runif(n * nn)))
x <- rnorm(n * nn) + sqrt(as.numeric(g))
xg <- split(x, g)
boxplot(xg, col = "lavender", notch = TRUE, varwidth = TRUE)
sapply(xg, length)
sapply(xg, mean)

### Calculate 'z-scores' by group (standardize to mean zero, variance one)
z <- unsplit(lapply(split(x, g), scale), g)

# or

zz <- x

```

```

split(zz, g) <- lapply(split(x, g), scale)

# and check that the within-group std dev is indeed one
tapply(z, g, sd)
tapply(zz, g, sd)

### data frame variation

## Notice that assignment form is not used since a variable is being added

g <- airquality$Month
l <- split(airquality, g)

## Alternative using a formula
identical(l, split(airquality, ~ Month))

l <- lapply(l, transform, Oz.Z = scale(Ozone))
aq2 <- unsplit(l, g)
head(aq2)
with(aq2, tapply(Oz.Z, Month, sd, na.rm = TRUE))

### Split a matrix into a list by columns
ma <- cbind(x = 1:10, y = (-4:5)^2)
split(ma, col(ma))

split(1:10, 1:2)

```

sprintf

sprintf

```

sprintf(fmt, ...)
gettextf(fmt, ..., domain = NULL, trim = TRUE)

```

fmt

```

...          fmt
trimdomain   gettext

```

```

sprintfsprintfNAInf-InfNaN
gettextf
fmt"symbol""language"typeoffmt%.0s

```

```

fmt...%aAdifeEgGosxX%

```

```

dioxX oXxa-fdi01NA

```



```

f "[-]mmm.ddd"NaNInf
eE [-]m.ddde[+-]xx[-]m.dddE[+-]xx
gG %e%E%f%f, %e
aA [-]0xh.hhhp[+-]dNaNInf%axp%A
    h
s NA"NA"
% %

as.character sas.doublef, e, E, g, Glength(fmt) > 1
%

m.n mn
-
+

0
# xX0x0XeefgGgG

%1$99$
*
fmt%
%s
sprintf-0-0.000

fmt

sprintf

```

<https://developer.r-project.org/Portability.html>
<https://pubs.opengroup.org/onlinepubs/9699919799/functions/snprintf.html>
man sprintf

[formatC](#)
[paste](#)
[gettext](#)

```

## be careful with the format: most things in R are floats
## only integer-valued reals get coerced to integer.

sprintf("%s is %f feet tall\n", "Sven", 7.1)      # OK
try(sprintf("%s is %i feet tall\n", "Sven", 7.1)) # not OK
  sprintf("%s is %i feet tall\n", "Sven", 7 ) # OK

## use a literal % :

sprintf("%.0f%% said yes (out of a sample of size %.0f)", 66.666, 3)

## various formats of pi :

sprintf("%f", pi)
sprintf("%.3f", pi)
sprintf("%1.0f", pi)
sprintf("%5.1f", pi)
sprintf("%05.1f", pi)
sprintf("%+f", pi)
sprintf("% f", pi)
sprintf("%-10f", pi) # left justified
sprintf("%e", pi)
sprintf("%E", pi)
sprintf("%g", pi)
sprintf("%g", 1e6 * pi) # -> exponential
sprintf("%.9g", 1e6 * pi) # -> "fixed"
sprintf("%G", 1e-6 * pi)

## no truncation:
sprintf("%1.f", 101)

## re-use one argument three times, show difference between %x and %X
xx <- sprintf("%1$d %1$x %1$X", 0:15)
xx <- matrix(xx, dimnames = list(rep("", 16), "%d%x%X"))
noquote(format(xx, justify = "right"))

## More sophisticated:

sprintf("min 10-char string '%10s'",
  c("a", "ABC", "and an even longer one"))

n <- 1:18
sprintf(paste0("e with %2d digits = %.", n, "g"), n, exp(1))

## Platform-dependent bad example: may pad with spaces or zeroes
sprintf("%09s", month.name)

## Using arguments out of order
sprintf("second %2$1.0f, first %1$5.2f, third %3$1.0f", pi, 2, 3)

## Using asterisk for width or precision
sprintf("precision %. *f, width '%*.3f'", 3, pi, 8, pi)

## Asterisk and argument re-use, 'e' example reiterated:
sprintf("e with %1$d digits = %2$. *1$g", n, exp(1))

```

```
## re-cycle arguments
sprintf("%s %d", "test", 1:3)

## binary output showing rounding/representation errors
x <- seq(0, 1.0, 0.1); y <- c(0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1)
cbind(x, sprintf("%a", x), sprintf("%a", y))
```

sQuote

```
sQuote(x, q = getOption("useFancyQuotes"))
dQuote(x, q = getOption("useFancyQuotes"))
```

```
x
q
```

```
qoptionsuseFancyQuotesFALSETRUE"UTF-8""TeX"
```

```
x
```

<https://www.cl.cam.ac.uk/~mgk25/ucs/quotes.html>

shQuote

```
op <- options("useFancyQuotes")
paste("argument", sQuote("x"), "must be non-zero")
options(useFancyQuotes = FALSE)
cat("\ndistinguish plain", sQuote("single"), "and",
    dQuote("double"), "quotes\n")
options(useFancyQuotes = TRUE)
cat("\ndistinguish fancy", sQuote("single"), "and",
    dQuote("double"), "quotes\n")
options(useFancyQuotes = "TeX")
```

```

cat("\ndistinguish TeX", sQuote("single"), "and",
    dQuote("double"), "quotes\n")
if(l10n_info()$`Latin-1`) {
  options(useFancyQuotes = c("\xab", "\xbb", "\xbf", "?"))
  cat("\n", sQuote("guillemet"), "and",
      dQuote("Spanish question"), "styles\n")
} else if(l10n_info()$`UTF-8`) {
  options(useFancyQuotes = c("\xc2\xab", "\xc2\xbb", "\xc2\xbf", "?"))
  cat("\n", sQuote("guillemet"), "and",
      dQuote("Spanish question"), "styles\n")
}
options(op)

```

srcfile

"srcref"[options](#)(keep.source = TRUE)

```

srcfile(filename, encoding = getOption("encoding"), Enc = "unknown")
srcfilecopy(filename, lines, timestamp = Sys.time(), isFile = FALSE)
srcfilealias(filename, srcfile)
getSrcLines(srcfile, first, last)
srcref(srcfile, lloc)
## S3 method for class 'srcfile'
print(x, ...)
## S3 method for class 'srcfile'
summary(object, ...)
## S3 method for class 'srcfile'
open(con, line, ...)
## S3 method for class 'srcfile'
close(con, ...)
## S3 method for class 'srcref'
print(x, useSource = TRUE, ...)
## S3 method for class 'srcref'
summary(object, useSource = FALSE, ...)
## S3 method for class 'srcref'
as.character(x, useSource = TRUE, to = x, ...)
.isOpen(srcfile)

```

```

filename
encoding
Enc          encodingparse
lines
timestamp
isFile       srcfilecopy
srcfile      srcfile
firstlastline

```

```
lloc
xobjectcon
useSource      srcfilesrcref
to             srcref
...
```

```
srcfilesrcrefsrcfilefileiconvlist
srcfilecopysrcfilecopyisFile
srcfilealiasrcfilealiasrcfile#line
getSrcLinesrcfile
srcrefsrcrefsrcrefsrcfilelloc
```

```
c(first_line, first_byte, last_line, last_byte, first_column,
  last_column, first_parsed, last_parsed)
```

```
#line
printsummaryopenclasesrcfilesrcrefsrcfilecopyopenfile
printsummaryas.charactersrcrefas.characterertosrcrefsrcrefsrcfilecopysrcref<srcref:
"file" chars 1:1 to 2:10>line:columnsummary
srcref"srcref"srcrefprint.defaultsrcrefprintuseSource = FALSEsrcrefuseSource =
FALSE<srcref: ....>
.is0pensrcfile
```

```
srcfilesrcrefsrcfile
srcfilecopysrcfilecopy
getSrcLines
srcrefsrcref
```

```
getSrcFilenameremoveSource
```

```
src <- srcfile(system.file("DESCRIPTION", package = "base"))
summary(src)
getSrcLines(src, 1, 4)
ref <- srcref(src, c(1, 1, 2, 1000))
ref
print(ref, useSource = FALSE)
```

StackOverflows

[options\("expressions"\)](#)[stackOverflowError](#)[errorcondition](#)

[CStackOverflowError](#)[usage](#)
[protectStackOverflowError](#)
[nodeStackOverflowError](#)
[expressionStackOverflowError](#)[options\("expressions"\)](#)

[tryCatch\(\)](#)[withCallingHandlers\(\)](#)[tryCatch\("error"\)](#)

[Cstack_info](#)

[Memoryoptions](#)

standardGeneric

[standardGeneric](#)

[standardGeneric\(f, fdef\)](#)

[f](#)

[fdef](#)

[standardGenericf](#)

[fdef](#)

[GenericFunctions](#)

startsWith

xprefixsuffix

```
startsWith(x, prefix)
endsWith(x, suffix)
```

x [character](#)
prefixsuffix [character](#)

```
startsWith()

substring(x, 1, nchar(prefix)) == prefix
```

```
grepl("^<prefix>", x)

prefixgreplx
prefixsuffix
```

[logical](#)xprefixsuffix

[grepl](#)[substring](#)[charmatch](#)[pmatch](#)

```
startsWith(search(), "package:") # typically at least two FALSE, nowadays often three
```

```
x1 <- c("Foobar", "bla bla", "something", "another", "blu", "brown",
       "blau blüht der Enzian")# non-ASCII
x2 <- cbind(
  startsWith(x1, "b"),
  startsWith(x1, "bl"),
  startsWith(x1, "bla"),
  endsWith(x1, "n"),
  endsWith(x1, "an"))
rownames(x2) <- x1; colnames(x2) <- c("b", "bl", "bla", "n", "an")
x2
```

```
## Non-equivalence in case of missing values in 'x', see Details:
x <- c("all", "but", NA_character_)
cbind(startsWith(x, "a"),
      substring(x, 1L, 1L) == "a",
      grepl("^a", x))
```

Startup

```
--no-envIRON/etc/RenvIRON.siteR_ENVIRON_USER.RenvIRON
--no-site-fileR_PROFILE/etc/Rprofile.sitelocal.Library.site.libPaths().First
.Last
--no-init-fileR_PROFILE_USER.Rprofile
utils::dump.frames
.RData--no-restore-data--no-restore
.First.First().First.sys(requireoptions("defaultPackages").OptRequireMethods()
.First.Last.RprofileRprofile.site.RDataoptions.RprofileRprofile.site
options(defaultPackages = " "character())R_DEFAULT_PACKAGES=NULL
options(defaultPackages = " ")R_DEFAULT_PACKAGES=""
R_HISTFILE.Rhistory--no-restore-history--no-restore
--vanilla--no-site-file--no-init-file--no-envIRON CMD--no-restore
```

```
#=${foo-bar}foobar${foo:-bar}foobar${foo}""bar${foo-${bar-blah}}$HOME
```

```
"${HOME}"PATH"${PATH}"/"
```

```
RenvIRON.siteRprofile.site/etc/i386/RenvIRON.site.RenvIRON.i386.RenvIRON
```

```
/etc/RenvIRON/etc/RenvIRON/etc/RenvIRON.site
```

```
R.app
```

```
R CMD checkR CMD buildRenvIRONR_CHECK_ENVIRONR_BUILD_ENVIRONRenvIRON
```

```
~/R/check.RenvIRON~/R/build.RenvIRON
```

```
~/RenvIRON~/RprofileR CMD checkR CMD buildR_ENVIRON_USERR_PROFILE_USER""
```

```
rw-FAQSys.getenv("R_USER")
```

```
.LastcommandArgs
```

```
X11quartz
```

```
readRenvIRON.RenvIRON
```

```
Rprof
```



```

## Not run:
## Example ~/.Renvi on Unix
R_LIBS=~/.R/library
PAGER=/usr/local/bin/less

## Example .Renvi on Windows
R_LIBS=C:/R/library
MY_TCLTK="c:/Program Files/Tcl/bin"
# Variable expansion in double quotes, string literals with backslashes in
# single quotes.
R_LIBS_USER="${APPDATA}"\R-library'

## Example of setting R_DEFAULT_PACKAGES (from R CMD check)
R_DEFAULT_PACKAGES='utils,grDevices,graphics,stats'
# this loads the packages in the order given, so they appear on
# the search path in reverse order.

## Example of .Rprofile
options(width=65, digits=5)
options(show.signif.stars=FALSE)
setHook(packageEvent("grDevices", "onLoad"),
        function(...) grDevices::ps.options(horizontal=FALSE))
set.seed(1234)
.First <- function() cat("\n Welcome to R!\n\n")
.Last <- function() cat("\n Goodbye!\n\n")

## Example of Rprofile.site
local({
  # add MASS to the default packages, set a CRAN mirror
  old <- getOption("defaultPackages"); r <- getOption("repos")
  r["CRAN"] <- "http://my.local.cran"
  options(defaultPackages = c(old, "MASS"), repos = r)
  ## (for Unix terminal users) set the width from COLUMNS if set
  cols <- Sys.getenv("COLUMNS")
  if(nzchar(cols)) options(width = as.integer(cols))
  # interactive sessions get a fortune cookie (needs fortunes package)
  if (interactive())
    fortunes::fortune()
})

## if .Renvi contains
FOOBAR="coo\bar"doh\ex"abc\def'"

## then we get
# > cat(Sys.getenv("FOOBAR"), "\n")
# coo\bardoh\exabc"def'

## End(Not run)

```

stop

stop
geterrmessage

```
stop(..., call. = TRUE, domain = NULL)
geterrmessage()
```

```
...
```

```
call.
```

```
domain          gettextNA
```

```
options(error=)signalCondition()options("show.error.messages")NULLq("no",
status = 1, runLast = FALSE)getOption("catch.script.errors")
```

```
geterrmessage()traceback()
```

```
getOption("warning.length")
```

```
geterrmessage"\n"
```

```
domain = NA...gettextf()
```

```
warningtryoptionsstopifnottryCatchwithCallingHandlers
```

```
gettext
```

```
iter <- 12
try(if(iter > 10) stop("too many iterations"))
```

```
tst1 <- function(...) stop("dummy error")
try(tst1(1:10, long, calling, expression))
```

```
tst2 <- function(...) stop("dummy error", call. = FALSE)
try(tst2(1:10, longcalling, expression, but.not.seen.in.Error))
```

stopifnot

...exprsallTRUEstopall

stopifnot(..., exprs, exprObject, local = TRUE)

```
...exprs      TRUE...exprs
              {
                expr1
                expr2
                ....
              }
              TRUETRUEif(.)
              ...
exprObject    exprs...expressioncallnameTRUE
local         exprsenvironmentstopifnot()
```

stopifnot(A, B)stopifnot(exprs= {A ; B})

```
{ if(any(is.na(A)) || !all(A)) stop(...);
  if(any(is.na(B)) || !all(B)) stop(...) }
```

stopifnot()tryCatch()sys.call()

```
stopifnot(exprs = { ... }),
...all.equalall.equal(*)"all.equal"pmatch()all.equalShowall.equal
```

NULL...TRUE

stopifnot(exprs = ..)

```
assertWRONG <- function(exprs) stopifnot(exprs = exprs)
```

stopifnot()exprsexprsexprsexprstopifnot()

```
assert <- function(exprs) eval.parent(substitute(stopifnot(exprs = exprs)))
```

stopifnot(exprs = *)

[stopwarningassertConditionstopifnot\(\)](#)

```
## NB: Some of these examples are expected to produce an error. To
##     prevent them from terminating a run with example() they are
##     piped into a call to try().

stopifnot(1 == 1, all.equal(pi, 3.14159265), 1 < 2) # all TRUE

m <- matrix(c(1,3,3,1), 2, 2)
stopifnot(m == t(m), diag(m) == rep(1, 2)) # all(.) |> TRUE

stopifnot(length(10)) |> try() # gives an error: '1' is *not* TRUE
## even when   if(1) "ok"   works

stopifnot(all.equal(pi, 3.141593), 2 < 2, (1:10 < 12), "a" < "b") |> try()
## More convenient for interactive "line by line" evaluation:
stopifnot(exprs = {
  all.equal(pi, 3.1415927)
  2 < 2
  1:10 < 12
  "a" < "b"
}) |> try()

eObj <- expression(2 < 3, 3 <= 3:6, 1:10 < 2)
stopifnot(exprObject = eObj) |> try()
stopifnot(exprObject = quote(3 == 3))
stopifnot(exprObject = TRUE)

# long all.equal() error messages are abbreviated:
stopifnot(all.equal(rep(list(pi),4), list(3.1, 3.14, 3.141, 3.1415))) |> try()

# The default error message can be overridden to be more informative:
m[1,2] <- 12
stopifnot("m must be symmetric" = m == t(m)) |> try()
#=> Error: m must be symmetric

##' warnifnot(): a "only-warning" version of stopifnot()
##' {Yes, learn how to use do.call(substitute, ...) in a powerful manner !!}
warnifnot <- stopifnot ; N <- length(bdy <- body(warnifnot))
bdy      <- do.call(substitute, list(bdy, list(stopifnot = quote(warnifnot))))
bdy[[N-1]] <- do.call(substitute, list(bdy[[N-1]], list(stop = quote(warning))))
body(warnifnot) <- bdy
warnifnot(1 == 1, 1 < 2, 2 < 2) # => warns " 2 < 2 is not TRUE  "
warnifnot(exprs = {
  1 == 1
  3 < 3  # => warns "3 < 3 is not TRUE"
})
```

strptime

```
"POSIXlt""POSIXct"
```

```
## S3 method for class 'POSIXct'
format(x, format = "", tz = "", usetz = FALSE, ...)
## S3 method for class 'POSIXlt'
format(x, format = "", usetz = FALSE,
       digits = getOption("digits.secs"), ...)

## S3 method for class 'POSIXt'
as.character(x, digits = if(inherits(x, "POSIXlt")) 14L else 6L,
            OutDec = ".", ...)

strptime(x, format = "", tz = "", usetz = FALSE, ...)
strptime(x, format, tz = "")
```

```
x          strptime"POSIXlt"strptime
tz          as.POSIXlt""GMT"
format      format"%Y-%m-%d                %H:%M:%S""%Y-%m-%d"digitsNULL
            options("digits.secs")"%OS<n>""%S"
...         strptime()digitsformat(<POSIXlt>)
usetz       "%Z"
digits      format()format()as.character()as.character()
OutDec      getOption("OutDec")
```

```
formatas.characterstrptime"POSIXlt""POSIXct"character
strptime"POSIXlt"xas.character
strptimeformat.POSIXltformat.POSIXct"POSIXlt"as.POSIXlt"Date"as.POSIXlt()-Inf
InfNaNaNNAformat()POSIXltdouble
xformat
%%x%LC_TIME$Sys.setlocaleLC_TIMELC_ALL
%OE%%
%a
%A
%b
%B
%c "%a %b %e %H:%M:%S %Y"
%C
%d
%D %m/%d/%y
%e
```

```

%F
%g %V
%G %V
%h %b
%H 24:00:00
%I
%j
%m
%M
%n
%p %I%H
    AM/PMa.m./p.m.
    %P%pP
%r %I:%M:%S %p
%R %H:%M
%S
%t
%T %H:%M:%S
%u
%U
%V %G%g%V
%w
%W
%x "%y/%m/%d"
%X "%H:%M:%S"
%y
%Y https://en.wikipedia.org/wiki/0\_\(year\)
    0:9999
%Z -0800-1400+1400
%Z

C%a%b%h
%Z%Z
%n%t

%k
%l
%s
%+ %c"%a %b %e %H:%M:%S %Z %Y"

%O[dHImMUVwWy]%E[cCyYxX]
%OSn0 <= n <= 6%OSdigitsdigits = NULLdigits = 0
format=".. %OSn .."<<digits = ngetOption("digits.secs")strftime()
format(<POSIX.t>)
strptime%OS%S
%format

```

formatstrptimeNANA_character_
strptime"POSIXlt"isdst"tz != ""2010-02-30 08:00"NA

[https://en.wikipedia.org/wiki/0_\(year\)-45-045](https://en.wikipedia.org/wiki/0_(year)-45-045)

"%04Y"R_PAD_YEARS_BY_ZERO%04Y%_4Y%_Y

"POSIXct"
[Sys.timezone](#)-0800
%z+0000"POSIXct"
%ztz = "UTC"

strptimestrptime
strptimeglibcstrptime
strftimetzcode<https://www.iana.org/time-zones>
strftimewcsftimeformat

"2001-02-28""14:01:02"T
strptime%d%eNAtzone

strptimeas.POSIXctstrptime
%c"%a %b %e %H:%M:%S %Y"
strftimeformat

https://dotat.at/tmp/ISO_8601-2004_E.pdf<https://www.qsl.net/g1smd/isopdf.htm>
<https://www.iso.org/iso/iso8601>https://en.wikipedia.org/wiki/ISO_8601

strftimestrptime

```

## locale-specific version of date()
format(Sys.time(), "%a %b %d %X %Y %Z")

## time to sub-second accuracy (if supported by the OS)
format(Sys.time(), "%H:%M:%OS3")

## read in date info in format 'ddmmmyyyy'
## This will give NA(s) in some locales (especially non-English ones);
## setting the C locale as in the commented lines will overcome this on
## most systems.
## lct <- Sys.getlocale("LC_TIME"); Sys.setlocale("LC_TIME", "C")
x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
z <- strptime(x, "%d%b%Y")
## Sys.setlocale("LC_TIME", lct)
z
(chz <- as.character(z)) # same w/o TZ
## *here* (but not in general), the same as format():
stopifnot(exprs = {
  identical(chz, format(z))
  grepl("^1960-0[137]-[03][012]$", chz[!is.na(z)])
})

## read in date/time info in format 'm/d/y h:m:s'
dates <- c("02/27/92", "02/27/92", "01/14/92", "02/28/92", "02/01/92")
times <- c("23:03:20", "22:29:56", "01:03:30", "18:21:03", "16:56:26")
x <- paste(dates, times)
z2 <- strptime(x, "%m/%d/%y %H:%M:%S")
z2
## *here* (but not in general), the same as format():
stopifnot(identical(format(z2), as.character(z2)))

## time with fractional seconds (setting `tz = ..` for reproducible output)
z3 <- strptime("20/2/06 11:16:16.683", "%d/%m/%y %H:%M:%OS", tz = "UTC")
z3 # prints without fractional seconds by default, digits.sec = NULL ("= 0")
format(z3, digits = 3) # shows extra digits
format(z3, digits = 6) # still 3 digits: *not* showing trailing zeros
format(z3, format = "%Y-%m-%d %H:%M:%OS6") # *does* keep trailing zeros
op <- options(digits.secs = 3) # global option, the default for `digits`
z3 # shows the 3 extra digits
options(op)
as.character(z3) # ditto

## time zone names are not portable, but 'EST5EDT' comes pretty close.
## (but its interpretation may not be universal: see ?timezones)
z4 <- strptime(c("2006-01-08 10:07:52", "2006-08-07 19:33:02"),
  "%Y-%m-%d %H:%M:%S", tz = "EST5EDT")
z4
attr(z4, "tzone")
as.character(z4)
z4$sec[2] <- pi # "very" fractional seconds
as.character(z4) # shows full precision
format(z4) # no fractional sec
format(z4, digits=8) # shows only 6 (hard-wired maximum)
format(z4, digits=4)

```



```
## An RFC 5322 header (Eastern Canada, during DST)
## In a non-English locale the commented lines may be needed.

## prev <- Sys.getlocale("LC_TIME"); Sys.setlocale("LC_TIME", "C")
strptime("Tue, 23 Mar 2010 14:36:38 -0400", "%a, %d %b %Y %H:%M:%S %z")
## Sys.setlocale("LC_TIME", prev)

## Make sure you know what the abbreviated names are for you if you wish
## to use them for input (they are matched case-insensitively):
format(s1 <- seq.Date(as.Date('1978-01-01'), by = 'day', len = 7), "%a")
format(s2 <- seq.Date(as.Date('2000-01-01'), by = 'month', len = 12), "%b")

## Non-finite date-times :
format(as.POSIXct(Inf)) # "Inf" (was NA in R <= 4.1.x)
format(as.POSIXlt(c(-Inf, Inf, NaN, NA))) # were all NA
```

strrep

```
strrep(x, times)
```

x	as.character
times	x

```
xtimesxtimes
```

```
strrep("ABC", 2)
strrep(c("A", "B", "C"), 1 : 3)
## Create vectors with the given numbers of spaces:
strrep(" ", 1 : 5)
```

strsplit

xsplit

strsplit(x, split, fixed = FALSE, perl = FALSE, useBytes = FALSE)

x

split fixed = TRUEsplitxsplitx

fixed TRUEsplitperl

perl

useBytes TRUE"bytes"Encoding

splitsplit = NULLsplit = character(0)

split = character(0)split = ""

splitx

```
repeat {
  if the string is empty
    break.
  if there is a match
    add the string to the left of the match to the output.
    remove the match and all to the left of it.
  else
    add the string to the output.
    break.
}
```

""

"[[:<:]]"perl = TRUE"(?!^)[[:<:]]"

xix[i]

xsplitEncodingperl = TRUE, useBytes = FALSE

xsplit"bytes"Encoding"bytes""bytes"xsplit"bytes"useBytes = TRUE"bytes""unknown"
"bytes"iconv

fixedperluseBytes

pastegrepsubncharsubstrstartsWithendsWith

```
PCRE_use_JITperl = TRUE
```

```
noquote(strsplit("A text I want to display with spaces", NULL)[[1]])

x <- c(as = "asfef", qu = "qwerty", "yuiop", "b", "stuff.blah.yech")
# split x on the letter e
strsplit(x, "e")

unlist(strsplit("a.b.c", "."))
## [1] "" "" "" "" "" ""
## Note that 'split' is a regexp!
## If you really want to split on '.', use
unlist(strsplit("a.b.c", "[.]"))
## [1] "a" "b" "c"
## or
unlist(strsplit("a.b.c", ".", fixed = TRUE))

## a useful function: rev() for strings
strReverse <- function(x)
  sapply(lapply(strsplit(x, NULL), rev), paste, collapse = "")
strReverse(c("abc", "Statistics"))

## get the first names of the members of R-core
a <- readLines(file.path(R.home("doc"), "AUTHORS"))[-(1:8)]
a <- a[(0:2)-length(a)]
(a <- sub(".*", "", a))
# and reverse them
strReverse(a)

## Note that final empty strings are not produced:
strsplit(paste(c("", "a", ""), collapse="#"), split="#")[1]
# [1] "" "a"
## and also an empty string is only produced before a definite match:
strsplit("", " ")[1] # character(0)
strsplit(" ", " ")[1] # [1] ""
```

strtoi

strtol

strtoi(x, base = 0L)

x [as.character](#)

base

```
strtol  
base = 0Lx0xX0x0X81610  
10azAZ1035
```

[xNA_integer_](#)

[as.integer](#)

```
strtoi(c("0xff", "077", "123"))  
strtoi(c("ffff", "FFFF"), 16L)  
strtoi(c("177", "377"), 8L)
```

`strtrim`

```
strtrim(x, width)
```

x	as.character
width	x

[substr](#)

x

[Encoding](#)

```
strtrim(c("abcdef", "abcdef", "abcdef"), c(1,5,10))
```

structure

structure

structure(.Data, ...)

.Data

... tag = value

"factor"

".Dim"."Dimnames"."Names"."Tsp"."Label"."dim"."dimnames"."names"."tsp"."levels"

tag = NULLtag.Data

attributesattr

structure(1:6, dim = 2:3)

strwrap

strwrap(x, width = 0.9 * getOption("width"), indent = 0,
exdent = 0, prefix = "", simplify = TRUE, initial = prefix)

x as.character

width

indent

exdent

prefixinitial initial

simplify TRUExx

simplifyTRUE

```
## Read in file 'THANKS'.
x <- paste(readLines(file.path(R.home("doc"), "THANKS")), collapse = "\n")
## Split into paragraphs and remove the first three ones
x <- unlist(strsplit(x, "\n[ \t\n]*\n"))[-(1:3)]
## Join the rest
x <- paste(x, collapse = "\n\n")
## Now for some fun:
writelines(strwrap(x, width = 60))
writelines(strwrap(x, width = 60, indent = 5))
writelines(strwrap(x, width = 60, exdent = 5))
writelines(strwrap(x, prefix = "THANKS> "))

## Note that messages are wrapped AT the target column indicated by
## 'width' (and not beyond it).
## From an R-devel posting by J. Hosking <jh910@juno.com>.
x <- paste(sapply(sample(10, 100, replace = TRUE),
  function(x) substring("aaaaaaaaa", 1, x)), collapse = " ")
sapply(10:40,
  function(m)
    c(target = m, actual = max(nchar(strwrap(x, m)))))
```

subset

```
subset(x, ...)

## Default S3 method:
subset(x, subset, ...)

## S3 method for class 'matrix'
subset(x, subset, select, drop = FALSE, ...)

## S3 method for class 'data.frame'
subset(x, subset, select, drop = FALSE, ...)

x
subset
select
drop          [
...
```

```
x[subset & !is.na(subset)]
```

```
subsetsubset
```

```
select
```

```
drop
```

```
droplevels
```

```
x
```

```
[subset
```

```
[transformdroplevels
```

```
subset(airquality, Temp > 80, select = c(Ozone, Temp))
```

```
subset(airquality, Day == 1, select = -Temp)
```

```
subset(airquality, select = Ozone:Wind)
```

```
with(airquality, subset(Ozone, Temp > 80))
```

```
## sometimes requiring a logical 'subset' argument is a nuisance
```

```
nm <- rownames(state.x77)
```

```
start_with_M <- nm %in% grep("^M", nm, value = TRUE)
```

```
subset(state.x77, start_with_M, Illiteracy:Murder)
```

```
# but in recent versions of R this can simply be
```

```
subset(state.x77, grepl("^M", nm), Illiteracy:Murder)
```

```
substitute
```

```
substituteexprenv
```

```
quote
```

```
enquoteFoo(...)  
quote(Foo(...))  
call
```

```
substitute(expr, env)
```

```
quote(expr)
```

```
enquote(cl)
```

```

expr
cl          callclassmode"call"
env

```

```

substitutemyplotdeparsesubstitutemyplot
envdelayedAssign()env.GlobalEnv
quotesubstitute

```

```

mode"call""name"

```

```

substitute
expression(...)expressioneval

```

```

missingbquotesQuotedQuoteQuotes'`"
all.names

```

```

require(graphics)
(s.e <- substitute(expression(a + b), list(a = 1))) #> expression(1 + b)
(s.s <- substitute( a + b,          list(a = 1))) #> 1 + b
c(mode(s.e), typeof(s.e)) # "call", "language"
c(mode(s.s), typeof(s.s)) # (the same)
# but:
(e.s.e <- eval(s.e))      #> expression(1 + b)
c(mode(e.s.e), typeof(e.s.e)) # "expression", "expression"

```

```

substitute(x <- x + 1, list(x = 1)) # nonsense

```

```

myplot <- function(x, y)
  plot(x, y, xlab = deparse1(substitute(x)),
       ylab = deparse1(substitute(y)))

```

```

## Simple examples about lazy evaluation, etc:

```

```

f1 <- function(x, y = x)      { x <- x + 1; y }
s1 <- function(x, y = substitute(x)) { x <- x + 1; y }
s2 <- function(x, y) { if(missing(y)) y <- substitute(x); x <- x + 1; y }
a <- 10
f1(a) # 11
s1(a) # 11
s2(a) # a
typeof(s2(a)) # "symbol"

```

substr

```
substr(x, start, stop)
substring(text, first, last = 1000000L)

substr(x, start, stop) <- value
substring(text, first, last = 1000000L) <- value
```

```
xtext
startfirst
stoplast
value
```

```
substringfirstlaststartstop
start""stop
xtextas.character
start
NANA
Encoding
"bytes"Encoding
```

```
substrxstartstop
substringxx
xtextattributesnames
xtextEncoding
```

```
substring<-last
ncharstrtrimnchar(type = "chars")
```

```
substring
```

```
startsWithendsWithstrsplitpastenchar
```

```

substr("abcdef", 2, 4)
substring("abcdef", 1:6, 1:6)
## strsplit() is more efficient ...

substr(rep("abcdef", 4), 1:4, 4:5)
x <- c("asfef", "qwerty", "yuioP", "b", "stuff.blah.yech")
substr(x, 2, 5)
substring(x, 2, 4:6)

X <- x
names(X) <- LETTERS[seq_along(x)]
comment(X) <- noquote("is a named vector")
str(aX <- attributes(X))
substring(x, 2) <- c("..", "+++")
substring(X, 2) <- c("..", "+++")
X
stopifnot(x == X, identical(aX, attributes(X)), nzchar(comment(X)))

```

sum

sum

sum(..., na.rm = FALSE)

...

na.rm TRUEFALSENaN

[Summary...](#)

na.rmFALSENaNaNNaNaNNaNaN

NULLinteger(0)

...integerdoubledouble

[Summary](#)x, ..., na.rm

sum

colSums

```
## Pass a vector to sum, and it will add the elements together.
sum(1:5)

## Pass several numbers to sum, and it also adds the elements.
sum(1, 2, 3, 4, 5)

## In fact, you can pass vectors into several arguments, and everything gets added.
sum(1:2, 3:5)

## If there are missing values, the sum is unknown, i.e., also missing, ....
sum(1:5, NA)
## ... unless we exclude missing values explicitly:
sum(1:5, NA, na.rm = TRUE)
```

summary

summary[methods](#)[class](#)

```
summary(object, ...)
```

Default S3 method:

```
summary(object, ..., digits, quantile.type = 7)
```

S3 method for class 'data.frame'

```
summary(object, maxsum = 7,
         digits = max(3, getOption("digits")-3), ...)
```

S3 method for class 'factor'

```
summary(object, maxsum = 100, ...)
```

S3 method for class 'matrix'

```
summary(object, ...)
```

S3 method for class 'summaryDefault'

```
format(x, digits = max(3L, getOption("digits") - 3L), zdigits = 4L, ...)
```

S3 method for class 'summaryDefault'

```
print(x, digits = max(3L, getOption("digits") - 3L), zdigits = 4L, ...)
```

object

x	summary()
maxsum	factor
digits	NULL signif() summary.default format() summary.data.frame summary.default missing(.) signif() print format

```

zdigits      zapsmall(*, digits = digits + zdigits)
quantile.type quantile(*, type=quantile.type)
...

```

```

factor maxsum - 1 "(Other)" maxsum
digits NULL "summaryDefault" format()
summary.lm summary.glm lm glm

```

```

summary
c("summaryDefault", "table") format print factor
"table" summary

```

```

anova summary.glm summary.lm

```

```

summary(attenu, digits = 4) #-> summary.data.frame(...), default precision
summary(attenu $ station, maxsum = 20) #-> summary.factor(...)

```

```

lst <- unclass(attenu$station) > 20 # logical with NAs
## summary.default() for logicals -- different from *.factor:
summary(lst)
summary(as.factor(lst))

```

```

## show the effect of zdigits for skewed data
set.seed(17); x <- rlnorm(100, sdlog=2)
dput(sx <- summary(x))
sx # exponential format for this data
print(sx, zdigits = 3) # fixed point: "more readable"

```

svd

```

svd(x, nu = min(n, p), nv = min(n, p), LINPACK = FALSE)

```

```

La.svd(x, nu = min(n, p), nv = min(n, p))

```

```

x
nu      0n = nrow(x)
nv      0p = ncol(x)
LINPACK

```

```

svdLa.svd
nu <= min(n, p)nv <= min(n, p)
1
NaNx

```

$$X = UDV',$$

$$UVV'DD_{ii}D = U'XV$$

```

d          xmin(n, p)
u          xnu > 0c(n, nu)
v          xnv > 0c(p, nv)

```

```
La.svdvvtv
```

```
DGESDDZGESDD
```

<https://netlib.org/lapack/>

https://netlib.org/lapack/lug/lapack_lug.html

eigenqr

```

hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, `+`) }
X <- hilbert(9)[, 1:6]
(s <- svd(X))
D <- diag(s$d)
s$u %*% D %*% t(s$v) # X = U D V'
t(s$u) %*% X %*% s$v # D = U' X V

```

sweep

```
sweep(x, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)
```

```
x
MARGIN          xSTATSx
STATS
FUN
check.margin    TRUESTATSxFALSE
...             FUN
```

```
FUNmatch.fun
```

```
FUNxSTATSaperm
```

```
STATSMARGINxSTATSsweep(x, MARGIN, as.array(STATS))STATS
```

```
x
```

[apply](#)[sweep](#)[scale](#)

```
require(stats) # for median
med.att <- apply(attitude, 2, median)
sweep(data.matrix(attitude), 2, med.att) # subtract the column medians
```

```
## More sweeping:
A <- array(1:24, dim = 4:2)
```

```
## no warnings in normal use
sweep(A, 1, 5)
(A.min <- apply(A, 1, min)) # == 1:4
sweep(A, 1, A.min)
sweep(A, 1:2, apply(A, 1:2, median))
```

```
## warnings when mismatch
sweep(A, 1, 1:3) # STATS does not recycle
sweep(A, 1, 6:1) # STATS is longer
```

```
## exact recycling:
sweep(A, 1, 1:2) # no warning
sweep(A, 1, as.array(1:2)) # warning

## Using named dimnames

dimnames(A) <- list(fee=1:4, fie=1:3, fum=1:2)

mn_fum_fie <- apply(A, c("fum", "fie"), mean)
mn_fum_fie
sweep(A, c("fum", "fie"), mn_fum_fie)
```

switch

switchEXPR...

switch(EXPR, ...)

EXPR

... EXPR

switch

EXPRfactor`nargs()`-1...3

EXPR...switch("cc", a = 1, cc =, cd =, d = 2)2...

EXPR

...NULL

`invisible`

switchEXPR

```

require(stats)
centre <- function(x, type) {
  switch(type,
    mean = mean(x),
    median = median(x),
    trimmed = mean(x, trim = .1))
}
x <- rcauchy(10)
centre(x, "mean")
centre(x, "median")
centre(x, "trimmed")

ccc <- c("b","QQ","a","A","bb")
# note: cat() produces no output for NULL
for(ch in ccc)
  cat(ch,":", switch(EXPR = ch, a = 1, b = 2:3), "\n")
for(ch in ccc)
  cat(ch,":", switch(EXPR = ch, a =, A = 1, b = 2:3, "Otherwise: last"), "\n")

## switch(f, *) with a factor f
ff <- gl(3,1, labels=LETTERS[3:1])
ff[1] # C
## so one might expect " is C" here, but
switch(ff[1], A = "I am A", B="Bb..", C=" is C")# -> "I am A"
## so we give a warning

## Numeric EXPR does not allow a default value to be specified
## -- it is always NULL
for(i in c(-1:3, 9)) print(switch(i, 1, 2 , 3, 4))

## visibility
switch(1, invisible(pi), pi)
switch(2, invisible(pi), pi)

```

Syntax

```

:: :::
$ @
[ [[
^
- +
:
%any% |>      %%%/%
* /
+ -

```



```

< > <= >= == !=
!
& &&
| ||
~
-> ->>
<- <<-
=
?

```

```
=
```

```
:::::$@
```

```
?+ -& && | ||
```

ArithmeticComparisonControlExtractLogicNumericConstantsParenQuotesReserved

```

## Logical AND ("&&") has higher precedence than OR ("||"):
TRUE || TRUE && FALSE # is the same as
TRUE || (TRUE && FALSE) # and different from
(TRUE || TRUE) && FALSE

```

```

## Special operators have higher precedence than "!" (logical NOT).
## You can use this for %in% :
! 1:10 %in% c(2, 3, 5, 7) # same as !(1:10 %in% c(2, 3, 5, 7))
## but we strongly advise to use the "!( ... )" form in this case!

```

```

## '=' has lower precedence than '<-' ... so you should not mix them
## (and '<-' is considered better style anyway):
## Not run: ## Consequently, this gives a ("non-catchable") error
x <- y = 5 #-> Error in (x <- y) = 5 : ....

```

```
## End(Not run)
```

Sys.getenv

Sys.getenv

```
Sys.getenv(x = NULL, unset = "", names = NA)
```

```
x          NULL
```

```
unset
```

```
names      NA
```

```
unset = NA""
```

```
xnames == TRUEnamesxunset
```

```
Sys.getenv()===
```

```
xnames"Dlist"print
```

```
Sys.setenvSys.getlocalegetwd
```

```
## whether HOST is set will be shell-dependent e.g. Solaris' csh did not.  
Sys.getenv(c("R_HOME", "R_PAPERSIZE", "R_PRINTCMD", "HOST"))
```

```
s <- Sys.getenv() # *all* environment variables  
op <- options(width=111) # (nice printing)  
names(s) # all settings (the values could be very long)  
head(s, 12) # using the Dlist print() method
```

```
## Language and Locale settings -- but rather use Sys.getlocale()  
s[grep("^L(C|ANG)", names(s))]  
## typically R-related:  
s[grep("^_?R_", names(s))]  
options(op)# reset
```

Sys.getpid

```
Sys.getpid()
```

```
Sys.getpid()
```

```
## Show files opened from this R process
if(.Platform$OS.type == "unix") ## on Unix-alikes such Linux, macOS, FreeBSD:
  system(paste("lsof -p", Sys.getpid()))
```

Sys.glob

```
Sys.glob(paths, dirmark = FALSE)
```

```
paths
dirmark      /
```

```
globhttps://pubs.opengroup.org/onlinepubs/9699919799/functions/glob.html
*?[]
```

```
.Sys.glob("*.RData").RData.aa.RDataSys.glob("*.*)...
[[...]]!...
[A-Z]-
?*[]
```

```
paths
```

```
path.expand
```

```
Sys.glob(file.path(R.home(), "library", "*", "R", "*.rdx"))
```

Sys.info

Sys.info()

sysnameDarwinSunOS

Sys.info()R.versionreleaseversion

sysname

release

version

nodename

machine

login "unknown"

user "unknown"

effective_user "unknown"

udomain

uname(2)getlogin(2)getpwuid(getuid())getpwuid(geteuid())

loginusereffective_userudomain

releaseversionreleaseversionrelease

"4.17.11-200.fc28.x86_64" # Linux (Fedora)

"3.16.0-5-amd64" # Linux (Debian)

"17.7.0" # macOS 10.13.6

"5.11" # Solaris

system("whoami")system("id")-[Ggu][nr]whoamiid -un/usr/xpg4/bin/id -un

win.version

.PlatformR.versionsessionInfo()

Sys.info()

An alternative (and probably better) way to get the login name on Unix

Sys.getenv("LOGNAME")

Sys.localeconv

Sys.localeconv()

LC_NUMERIC.LC_NUMERICSys.setlocale

LC_MONETARY

NULL

Sys.setlocale

```
Sys.localeconv()
## The results in the C locale are
##   decimal_point   thousands_sep      grouping   int_curr_symbol
##   " ."           ""                  ""         ""
##   currency_symbol mon_decimal_point mon_thousands_sep   mon_grouping
##   ""              ""                  ""                 ""
##   positive_sign    negative_sign     int_frac_digits      frac_digits
##   ""               ""                 "127"                "127"
##   p_cs_precedes    p_sep_by_space    n_cs_precedes         n_sep_by_space
##   "127"            "127"              "127"                 "127"
##   p_sign_posn      n_sign_posn
##   "127"            "127"

## Now try your default locale (which might be "C").
old <- Sys.getlocale()
## The category may not be set:
## the following may do so, but it might not be supported.
Sys.setlocale("LC_MONETARY", locale = "")
Sys.localeconv()
## or set an appropriate value yourself, e.g.
Sys.setlocale("LC_MONETARY", "de_AT")
Sys.localeconv()
Sys.setlocale(locale = old)

## Not run: read.table("foo", dec=Sys.localeconv()[ "decimal_point"])
```

`sys.parent`

[environment](#)

```
sys.call(which = 0)
sys.frame(which = 0)
sys.nframe()
sys.function(which = 0)
sys.parent(n = 1)
```

```
sys.calls()
sys.frames()
sys.parents()
sys.on.exit()
sys.status()
parent.frame(n = 1)
```

`which`

`n`

```
.GlobalEnv sys.calls sys.functions sys.frame
sys.calls sys.functions sys.frame which which.GlobalEnv
sys.parent n n
sys.nframe
sys.call sys.frame sys.parents
sys.sys.status
sys.status() sys.call sys.parent sys.frame sys.status
sys.on.exit() on.exit
parent.frame(n) sys.frame(sys.parent(n))
```

```
sys.calls sys.functions sys.frame parent.frame
```

```
sys.parent parent.frame
```

`env` [eval](#)

```
parent.frame
```

```
evalsys.frameparent.frame
```

```
require(utils)
```

```
## Note: the first two examples will give different results
## if run by example().
ff <- function(x) gg(x)
gg <- function(y) sys.status()
str(ff(1))
```

```
gg <- function(y) {
  ggg <- function() {
    cat("current frame is", sys.nframe(), "\n")
    cat("parents are", sys.parents(), "\n")
    print(sys.function(0)) # ggg
    print(sys.function(2)) # gg
  }
  if(y > 0) gg(y-1) else ggg()
}
gg(3)
```

```
t1 <- function() {
  aa <- "here"
  t2 <- function() {
    ## in frame 2 here
    cat("current frame is", sys.nframe(), "\n")
    str(sys.calls()) ## list with two components t1() and t2()
    cat("parents are frame numbers", sys.parents(), "\n") ## 0 1
    print(ls(envir = sys.frame(-1))) ## [1] "aa" "t2"
    invisible()
  }
  t2()
}
t1()
```

```
test.sys.on.exit <- function() {
  on.exit(print(1))
  ex <- sys.on.exit()
  str(ex)
  cat("exiting...\n")
}
test.sys.on.exit()
## gives 'language print(1)', prints 1 on exit
```

```
## An example where the parent is not the next frame up the stack
## since method dispatch uses a frame.
```

```
as.double.foo <- function(x)
{
  str(sys.calls())
  print(sys.frames())
  print(sys.parents())
  print(sys.frame(-1)); print(parent.frame())
  x
}
```

```
t2 <- function(x) as.double(x)
a <- structure(pi, class = "foo")
t2(a)
```

Sys.readlink

readlink

Sys.readlink(paths)

paths [path.expand](#)

```
paths""NA
readlink""
```

[file.symlinkfile.info](#)

```
##' To check if files (incl. directories) are symbolic links:
is.symlink <- function(paths) isTRUE(nzchar(Sys.readlink(paths), keepNA=TRUE))
## will return all FALSE when the platform has no `readlink` system call.
is.symlink("/foo/bar")
```

Sys.setenv

Sys.setenv[Sys.getenv](#)
Sys.unsetenv

Sys.setenv(...)

Sys.unsetenv(x)

...
x


```
Sys.setenv="" Sys.setenv(F00 = "")F00
```

```
cmd.exe set
```

```
Sys.unsetenv
```

```
Sys.unsetenv""
```

```
Sys.getenv
```

```
setwd
```

```
Sys.setlocale Sys.setLanguage LANGUAGE conditionMessage
```

```
print(Sys.setenv(R_TEST = "testit", "A+C" = 123)) # `A+C` could also be used
Sys.getenv("R_TEST")
Sys.unsetenv("R_TEST") # on Unix-alike may warn and not succeed
Sys.getenv("R_TEST", unset = NA)
```

```
Sys.setFileTime
```

```
Sys.setFileTime(path, time)
```

```
path
```

```
time          "POSIXct"paths
```

```
utimensat utimes utime
```

```
SetFileTime
```

```
Sys.setFileTime path time
```

`Sys.sleep`

`Sys.sleep(time)`

`time`

`Ctrl-C`
`Esc`

`timeNANaNI`
`Inf`

`NULL`

`sleepSleep`

```
testit <- function(x)
{
  p1 <- proc.time()
  Sys.sleep(x)
  proc.time() - p1 # The cpu usage should be negligible
}
testit(3.7)
```

`sys.source`

```
sys.source(file, envir = baseenv(), chdir = FALSE,
  keep.source = getOption("keep.source.pkgs"),
  keep.parse.data = getOption("keep.parse.data.pkgs"),
  toplevel.env = as.environment(envir))
```

```

file
envir      baseenv()envir
chdir      TRUEfile
keep.source TRUEoptions(keep.source = *)
keep.parse.data
            TRUEkeep.sourceTRUEoptions(keep.parse.data = *)
toplevel.env

```

```

keep.source = FALSEkeep.parse.data = FALSE

```

```

envir

```

```

topenv()sys.source

```

```

sourceloadNamespacelibrary(.)sys.source(.)

```

```

## a simple way to put some objects in an environment
## high on the search path
tmp <- tempfile()
writelines("aaa <- pi", tmp)
env <- attach(NULL, name = "myenv")
sys.source(tmp, env)
unlink(tmp)
search()
aaa
detach("myenv")

```

Sys.time

```

Sys.timeSys.Date

```

```

Sys.time()
Sys.Date()

```

```

Sys.time
Sys.Date

```

```

Sys.time"POSIXct"
Sys.Date"Date"

```

```
Sys.time"POSIXct"format.POSIXct
```

```
date
```

```
Sys.timezone
```

```
system.time
```

```
Sys.time()  
## print with possibly greater accuracy:  
op <- options(digits.secs = 6)  
Sys.time()  
options(op)
```

```
## locale-specific version of date()  
format(Sys.time(), "%a %b %d %X %Y")
```

```
Sys.Date()
```

```
Sys.which
```

```
which
```

```
Sys.which(names)
```

```
names
```

```
which
```

```
.exe.com.cmd.batssystem
```

```
which/usr/bin/which
```

```
namesnames""""names
```

```
\
```

```
whichcsh
```

```
## the first two are likely to exist everywhere  
## texi2dvi exists on most Unix-alikes but not under MiKTeX  
Sys.which(c("ftp", "ping", "texi2dvi", "this-does-not-exist"))
```

system

systemcommand

```
system(command, intern = FALSE,
        ignore.stdout = FALSE, ignore.stderr = FALSE,
        wait = TRUE, input = NULL, show.output.on.console = TRUE,
        minimized = FALSE, invisible = TRUE, timeout = 0,
        receive.console.signals = wait)
```

command

intern NA

ignore.stdoutignore.stderr
 NAstdoutstderr

wait NAintern = TRUERgui

input command

timeout command

receive.console.signals

 NACtrl-Cintern = TRUEwait = TRUE

show.output.on.consoleminimizedinvisible

[system2](#)

commandshQuote/bin/shcommand;

systemshell

internTRUEpopencharacterinternFALSEsystem

wait&

timeoutwait = TRUE

/dev/nulltostopuser.childdsys.childproc_timeproc.time

system[Sys.which](#)

receive.console.signals = TRUEwait = FALSE

intern = TRUEcommand"status""errmsg"

intern = FALSE0127wait = TRUEwait = FALSE0

124

```

stderrignore.stderr = TRUE

  system("some command 2>&1", intern = TRUE)

stdoutstderrintern = FALSE

system

  systemcommandshellcommandcmd.exe
  systemsystem2shell
  stdoutstderrsystem(intern = TRUE)stderrignore.stderr = TRUE

  commandshQuote
  show.output.on.consoleminimizedinvisibleRgui

```

```

man systemman sh
.Platform
pipe

```

```

# list all files in the current directory using the -F flag
## Not run: system("ls -F")

# t1 is a character vector, each element giving a line of output from who
# (if the platform has who)
t1 <- try(system("who", intern = TRUE))

try(system("ls fizzlipuzzli", intern = TRUE, ignore.stderr = TRUE))
# zero-length result since file does not exist, and will give warning.

```

system.file

```

system.file(..., package = "base", lib.loc = NULL,
            mustWork = FALSE)

```

```

...
package
lib.loc      NULL
mustWork     TRUE

```

```

file.exists
...
find.packagelib.loc = NULL.libPaths().libPaths()

...""mustWork = TRUE

system.file()

R.homelist.files
Sys.glob

system.file()          # The root of the 'base' package
system.file(package = "stats") # The root of package 'stats'
system.file("INDEX")
system.file("help", "AnIndex", package = "splines")

```

```
system.time
```

```

expr

system.time(expr, gcFirst = TRUE)

expr
gcFirst      TRUE

system.timeproc.timeexprproc.timeproc.time
unix.timesystem.time
gcFirstTRUEgcexpr

"proc_time"proc.time

proc.timetime
setTimeLimit
Sys.time

```

```

require(stats)
system.time(for(i in 1:100) mad(runif(1000)))
## Not run:
exT <- function(n = 10000) {
  # Purpose: Test if system.time works ok;  n: loop size
  system.time(for(i in 1:n) x <- mean(rt(1000, df = 4)))
}
#-- Try to interrupt one of the following (using Ctrl-C / Escape):
exT()          #- about 4 secs on a 2.5GHz Xeon
system.time(exT())  #- +/- same

## End(Not run)

```

system2

system2command

```

system2(command, args = character(),
        stdout = "", stderr = "", stdin = "", input = NULL,
        env = character(), wait = TRUE,
        minimized = FALSE, invisible = TRUE, timeout = 0,
        receive.console.signals = wait)

```

command

```

args          commandshQuote
stdoutstderr  stdoutstderr""NULLFALSETRUE
stdin         ""input
input         command
env
wait          NAstdout = TRUEstderr = TRUERgui
timeout       command
receive.console.signals
              NActrl-cintern = TRUEwait = TRUE
minimizedinvisible

```

[systemcommandshQuote](#)

command[system](#)

envRmake

lsstdout = TRUEstdout = "some_file_name"

stderr = TRUEstdout = TRUE

timeoutwait = TRUE

/dev/nulltostop

receive.console.signals = TRUEwait = FALSE


```
stdout = TRUEstderr = TRUEcommand"status""errmsg"
0127wait = TRUEwait = FALSE0
124
```

```
system2systemcommandstdoutstderrsystemshell
stdoutstderrTRUE
```

```
system
```

```
t
```

```
data.framextx
```

```
t(x)
```

```
x
```

```
"data.frame"
as.matrixx
```

```
dimdimnamesx
```

```
 $AA^H A^* \text{Conj}(t(A))$ 
```

```
aperm
```

```
a <- matrix(1:30, 5, 6)
ta <- t(a) ##-- i.e., a[i, j] == ta[j, i] for all i,j :
for(j in seq(ncol(a)))
  if(! all(a[, j] == ta[j, ])) stop("wrong transpose")
```

table

table

```
table(...,
  exclude = if (useNA == "no") c(NA, NaN),
  useNA = c("no", "ifany", "always"),
  dnn = list.names(...), deparse.level = 1)
```

```
as.table(x, ...)
is.table(x)
```

```
## S3 method for class 'table'
as.data.frame(x, row.names = NULL, ...,
  responseName = "Freq", stringsAsFactors = TRUE,
  sep = "", base = list(LETTERS))
```

```
...          listas.tableas.data.frame
exclude      ...NAuseNAuseNA = "ifany"
useNA        NA
dnn
deparse.level dnn
x            "table"as.data.frameas.data.frame.table(x, *)xarray
row.names
responseName
stringsAsFactors

sepbase      provideDimnames
```

```
dnnlist.names...listnames()names...deparse.level = 0deparse.level = 1
deparse.level = 2
excludetable
useNANA"no""ifany""always"NAuseNAexcluded.patho
excludeuseNAfactoraddNA
afactor(a, exclude=exclude)NA
summary"table"tablextabschisq.test
```

```
table()"table"array
as.tableis.table
as.data.frame"table"responseNamextabs
```

tabulate

ftablemargin.tableprop.tableaddmargins

addNANA

xtabs

```
require(stats) # for rpois and xtabs
## Simple frequency distribution
table(rpois(100, 5))
## Check the design:
with(warpbreaks, table(wool, tension))
table(state.division, state.region)

# simple two-way contingency table
with(airquality, table(cut(Temp, quantile(Temp)), Month))

a <- letters[1:3]
table(a, sample(a))          # dnn is c("a", "")
table(a, sample(a), dnn = NULL) # dimnames() have no names
table(a, sample(a), deparse.level = 0) # dnn is c("", "")
table(a, sample(a), deparse.level = 2) # dnn is c("a", "sample(a)")

## xtabs() <-> as.data.frame.table() :
UCBAdmissions ## already a contingency table
DF <- as.data.frame(UCBAdmissions)
class(tab <- xtabs(Freq ~ ., DF)) # xtabs & table
## tab *is* "the same" as the original table:
all(tab == UCBAdmissions)
all.equal(dimnames(tab), dimnames(UCBAdmissions))

a <- rep(c(NA, 1/0:3), 10)
table(a)          # does not report NA's
table(a, exclude = NULL) # reports NA's
b <- factor(rep(c("A","B","C"), 10))
table(b)
table(b, exclude = "B")
d <- factor(rep(c("A","B","C"), 10), levels = c("A","B","C","D","E"))
table(d, exclude = "B")
print(table(b, d), zero.print = ".")

## NA counting:
is.na(d) <- 3:4
d. <- addNA(d)
d.[1:7]
table(d.) # "", exclude = NULL" is not needed
## i.e., if you want to count the NA's of 'd', use
table(d, useNA = "ifany")

## "pathological" case:
d.patho <- addNA(c(1,NA,1:2,1:3))[-7]; is.na(d.patho) <- 3:4
```

```

d.patho
## just 3 consecutive NA's ? --- well, have *two* kinds of NAs here :
as.integer(d.patho) # 1 4 NA NA 1 2
##
## In R >= 3.4.0, table() allows to differentiate:
table(d.patho) # counts the "unusual" NA
table(d.patho, useNA = "ifany") # counts all three
table(d.patho, exclude = NULL) # (ditto)
table(d.patho, exclude = NA) # counts none

## Two-way tables with NA counts. The 3rd variant is absurd, but shows
## something that cannot be done using exclude or useNA.
with(airquality,
     table(OzHi = Ozone > 80, Month, useNA = "ifany"))
with(airquality,
     table(OzHi = Ozone > 80, Month, useNA = "always"))
with(airquality,
     table(OzHi = Ozone > 80, addNA(Month)))

```

tabulate

tabulatebin

```
tabulate(bin, nbins = max(1, bin, na.rm = TRUE))
```

bin

nbins

tabulatetable

bin

binas.integer

integerdouble1, ..., nbinsNA

```
bin231length(bin) > .Machine$integer.maxbin
```

tablefactor

```

tabulate(c(2,3,5))
tabulate(c(2,3,3,5), nbins = 10)
tabulate(c(-2,0,2,3,3,5)) # -2 and 0 are ignored
tabulate(c(-2,0,2,3,3,5), nbins = 3)
tabulate(factor(letters[1:10]))

```

Tailcall	TailcallExec
----------	--------------

TailcallExec

Tailcall(FUN, ...)
Exec(expr, envir)

FUN
...
expr
envir exprExec

TailcallFUNExecexprprenvirTailcallExecon.exit
TailcallExec

[tracebacks](#)[sys.calls](#)TailcallExec

TailcallExec

[Recall](#)[force](#)

```
## tail-recursive log10-factorial
lfact <- function(n) {
  lfact_iter <- function(val, n) {
    if (n <= 0)
      val
    else {
      val <- val + log10(n) # forces val
      Tailcall(lfact_iter, val, n - 1)
    }
  }
  lfact_iter(0, n)
}
10 ^ lfact(3)
lfact(100000)

## simplified variant of do.call using Exec:
docall <- function (what, args, quote = FALSE) {
  if (!is.list(args))
    stop("second argument must be a list")
  if (quote)
    args <- lapply(args, enquote)
```

```

      Exec(as.call(c(list(substitute(what)), args)), parent.frame())
}
## the call stack does not contain the call to docall:
docall(function() sys.calls(), list()) |>
  Find(function(x) identical(x[[1]], quote(docall)), x = _)
## contrast to do.call:
do.call(function(x) sys.calls(), list()) |>
  Find(function(x) identical(x[[1]], quote(do.call)), x = _)

```

tapply

```
tapply(X, INDEX, FUN = NULL, ..., default = NA, simplify = TRUE)
```

X	split[
INDEX	listfactorXas.factorXfsplit
FUN	NULL+%%FUNNULLtapplytapply
...	FUN
default	array (default, dim = ..) array() NANANA_real_as.raw(0)"raw" FUN(integer(0))FUN = sum00L
simplify	FALSEtapply"list" listdimTRUEFUNtapply

FUNNULL[match.fun](#)

FUNtapplyFUNFUNmeanvarsimplifyTRUEtapplyNAINDEXnlevels()INDEX"Date"

simplify = TRUE

FUNtapply[listFUNdim](#)

[dimnames](#)INDEX

NULL

[array2DF](#)tapply

FUN...FUNX

[byaggregate](#)tapplyapplylapplysapplymapply

[array2DF](#)

```

require(stats)
groups <- as.factor(rbinom(32, n = 5, prob = 0.4))
tapply(groups, groups, length) #- is almost the same as
table(groups)

## contingency table from data.frame : array with named dimnames
tapply(warpbreaks$breaks, warpbreaks[,-1], sum)
tapply(warpbreaks$breaks, warpbreaks[, 3, drop = FALSE], sum)

n <- 17; fac <- factor(rep_len(1:3, n), levels = 1:5)
table(fac)
tapply(1:n, fac, sum)
tapply(1:n, fac, sum, default = 0) # maybe more desirable
tapply(1:n, fac, sum, simplify = FALSE)
tapply(1:n, fac, range)
tapply(1:n, fac, quantile)
tapply(1:n, fac, length) ## NA's
tapply(1:n, fac, length, default = 0) # == table(fac)

## example of ... argument: find quarterly means
tapply(presidents, cycle(presidents), mean, na.rm = TRUE)

ind <- list(c(1, 2, 2), c("A", "A", "B"))
table(ind)
tapply(1:3, ind) #-> the split vector
tapply(1:3, ind, sum)

## Some assertions (not held by all patch proposals):
nq <- names(quantile(1:5))
stopifnot(
  identical(tapply(1:3, ind), c(1L, 2L, 4L)),
  identical(tapply(1:3, ind, sum),
    matrix(c(1L, 2L, NA, 3L), 2, dimnames = list(c("1", "2"), c("A", "B")))),
  identical(tapply(1:n, fac, quantile)[-1],
    array(list(`2` = structure(c(2, 5.75, 9.5, 13.25, 17), names = nq),
      `3` = structure(c(3, 6, 9, 12, 15), names = nq),
      `4` = NULL, `5` = NULL), dim=4, dimnames=list(as.character(2:5)))))

```

taskCallback

```

addTaskCallback
removeTaskCallbackaddTaskCallback
taskCallbackManager

```

```

addTaskCallback(f, data = NULL, name = character())
removeTaskCallback(id)

```

```
f          data
data       f
id         getTaskCallbackNames\(\)addTaskCallback
name
```

```
expression1 ; expression2
dataaddTaskCallback
```

```
addTaskCallback
removeTaskCallbackFALSE
```

```
getTaskCallbackNametaskCallbackManager
https://developer.r-project.org/TaskHandlers.pdf
```

```
times <- function(total = 3, str = "Task a") {
  ctr <- 0
  function(expr, value, ok, visible) {
    ctr <-<= ctr + 1
    cat(str, ctr, "\n")
    keep.me <- (ctr < total)
    if (!keep.me)
      cat("handler removing itself\n")

    # return
    keep.me
  }
}

# add the callback that will work for
# 4 top-level tasks and then remove itself.
n <- addTaskCallback(times(4))

# now remove it, assuming it is still first in the list.
removeTaskCallback(n)

## See how the handler is called every time till "self destruction":

addTaskCallback(times(4)) # counts as once already

sum(1:10) ; mean(1:3) # two more
sinpi(1)             # 4th - and "done"
cospi(1)
tanpi(1)
```

taskCallbackManager

```
taskCallbackManager(handlers = list(), registered = FALSE,  
                    verbose = FALSE)
```

handlers	"f""data"add
registered	evaluateFALSEaddevaluateTRUE addTaskCallback
verbose	TRUE

[list](#)

add()	registerevaluate
remove()	
evaluate()	
suspend()	status
register()	evaluateadd
callbacks()	

<https://developer.r-project.org/TaskHandlers.pdf>

[addTaskCallbackremoveTaskCallbackgetTaskCallbackNames](#)

```
# create the manager  
h <- taskCallbackManager()  
  
# add a callback  
h$add(function(expr, value, ok, visible) {  
  cat("In handler\n")  
  return(TRUE)  
}, name = "simpleHandler")  
  
# look at the internal callbacks.  
getTaskCallbackNames()  
  
# look at the R-level callbacks  
names(h$callbacks())  
  
removeTaskCallback("R-taskCallbackManager")
```

taskCallbackNames

```
getTaskCallbackNames()
```

```
addTaskCallback
```

```
taskCallbackManager
```

```
addTaskCallbackremoveTaskCallbacktaskCallbackManager
```

```
https://developer.r-project.org/TaskHandlers.pdf
```

```
n <- addTaskCallback(function(expr, value, ok, visible) {  
  cat("In handler\n")  
  return(TRUE)  
}, name = "simpleHandler")
```

```
getTaskCallbackNames()
```

```
# now remove it by name  
removeTaskCallback("simpleHandler")
```

```
h <- taskCallbackManager()  
h$add(function(expr, value, ok, visible) {  
  cat("In handler\n")  
  return(TRUE)  
}, name = "simpleHandler")  
getTaskCallbackNames()  
removeTaskCallback("R-taskCallbackManager")
```

tempfile

```
tempfile
```

```
tempfile(pattern = "file", tmpdir = tempdir(), fileext = "")  
tempdir(check = FALSE)
```

```
pattern
tmpdir
fileext
check          logicalTRUEFALSEtmpdir()
```

```
tempfiletmpdir
tmpdirpatternfileext
tmpdirtmpdir()TMPDIRTMPTEMP/tmpdir()
```

```
tempfiletempfile
tmpdir
```

```
tmpdirnormalizePath
```

```
mclapplymakeForkClustertempfile
```

```
tmpdir()mkdtempglibc
RtmpXXXXXX.[A-Za-z0-9].
```

```
unlink
```

```
tempfile(c("ab", "a b c"))  # give file name with spaces in!

tempfile("plot", fileext = c(".ps", ".pdf"))

tmpdir() # works on all platforms with a platform-dependent result
```

```
## Show how 'check' is working on some platforms:
if(exists("I'm brave") && `I'm brave` &&
  identical(.Platform$OS.type, "unix") && grepl("^/tmp/", tmpdir())) {
  cat("Current tmpdir(): ", tmpdir(), "\n")
  cat("Removing it :", file.remove(tmpdir()),
      "; dir.exists(tmpdir()):", dir.exists(tmpdir()), "\n")
  cat("and now  tmpdir(check = TRUE) :", tmpdir(check = TRUE), "\n")
}
```

textConnection

```
textConnection(object, open = "r", local = FALSE,  
               name = deparse1(substitute(object)),  
               encoding = c("", "bytes", "UTF-8"))
```

```
textConnectionValue(con)
```

object	NULL
open	"r""w""a"
local	TRUE
name	character
encoding	object
con	

```
closeobject
```

```
localisIncomplete"\n"lockBindingclosetextConnectionValueobject = NULLEncoding  
mode = "a"  
seekseek
```

```
textConnection"textConnection""connection"  
textConnectionValue
```

```
file\(\)  
vsnprintf
```

```
connectionsshowConnectionspushBackcapture.output
```

```

zz <- textConnection(LETTERS)
readLines(zz, 2)
scan(zz, "", 4)
pushBack(c("aa", "bb"), zz)
scan(zz, "", 4)
close(zz)

zz <- textConnection("foo", "w")
writelines(c("testit1", "testit2"), zz)
cat("testit3 ", file = zz)
isIncomplete(zz)
cat("testit4\n", file = zz)
isIncomplete(zz)
close(zz)
foo

# capture R output: use part of example from help(lm)
zz <- textConnection("foo", "w")
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.5, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
sink(zz)
anova(lm.D9 <- lm(weight ~ group))
cat("\nSummary of Residuals:\n\n")
summary(resid(lm.D9))
sink()
close(zz)
cat(foo, sep = "\n")

```

tilde

y ~ model

ymodel

call[[~

formula

timezones

Sys.timezone

Sys.timezone(location = TRUE)

OlsonNames(tzdir = NULL)

location FALSE

tzdir

TZSys.timezone()TZTZTZ.Rprofile

TZUTCGMThttps://en.wikipedia.org/wiki/Coordinated_Universal_TimeSys.timezone

POSIXct

.sys.timezone

Sys.timezoneNAUTC"Europe/London"

"Europe/London"GBGB-EireEurope/BelfastEurope/GuernseyEurope/Isle_of_Man

Europe/JerseyPST8PDTAmerica/Los_Angeles

OlsonNames"Version""2023a"--with-internal-tzcodetzdata.zi

"UTC""GMT"

tzsetTZhttps://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html#tag_08

Europe/LondonAmerica/Los_AngelesPacific/EasterEST5EDTGBESTEST5EDT<https://data.iana.org/time-zones/tz-link.html>

US/EasternContinent/CityEtc/...

PMTWET/WESTCET/CEST

-03+0845"%z"strftime

OlsonNames/usr/share/zoneinfo/usr/share/lib/zoneinfozone1970.tabzone.tabEST5EDT

<https://en.wikipedia.org/wiki/Zone.tab>

--with-internal-tzcodefile.path(R.home("share"), "zoneinfo")VERSION

/var/db/timezone/zoneinfoTZDIRzoneinfo"internal""macOS"TZDIR

TZDIRglibc

TZtzas.POSIXltzoneinfo

<http://unicode.org/repos/cldr/trunk/common/supplemental/windowsZones.xml>TZ

Etc/GMT+nEtc/GMT-nEtc/PST8PDT

LMTGMT+0

tzselectsystemd timedatectl list-timezones

```
OlsonNamesbackwardbackzone
EST5EDTPST8PDTmusl
```

```
NA
```

```
TZ
```

```
tzcodehttps://www.iana.org/time-zonesglibcmusl-libclocaltime/etc/usr/local/etc
/usr/local/etc/zoneinfo localtime
```

```
/etc/localtime
```

```
localtime
```

```
/etc/localtime
```

```
systemd timedatectl
```

```
/etc/timezone
```

```
/etc/localtime Sys.timezone
```

```
"gmt"
```

```
Sys.time as.POSIXlt
```

```
https://en.wikipedia.org/wiki/Time\_zonehttps://data.iana.org/time-zones/tz-link.html
```

```
https://data.iana.org/time-zones/theory.html
```

```
Sys.timezone()
```

```
str(OlsonNames()) ## typically around six hundred names,
## typically some acronyms/aliases such as "UTC", "NZ", "MET", "Eire", ..., but
## mostly pairs (and triplets) such as "Pacific/Auckland"
table(sl <- grepl("/", OlsonNames()))
OlsonNames()[!sl] # the simple ones
head(0sl <- strsplit(OlsonNames()[sl], "/"))
(t0sl <- table(vapply(0sl, `[`, "", 1))) # Continents, countries, ...
table(lengths(0sl)) # most are pairs, some triplets
str(0sl[lengths(0sl) >= 3]) # "America" South and North ...
```

toString

[format](#)

```
toString(x, ...)
```

```
## Default S3 method:  
toString(x, width = NULL, ...)
```

```
x  
width          NULL  
...
```

```
widthnchar(type = "width")  
x", "widthNULLwidth - 4...width
```

[format](#)

```
x <- c("a", "b", "aaaaaaaaa")  
toString(x)  
toString(x, width = 8)
```

trace

trace[browserrecover](#)untrace

```
trace(what, tracer, exit, at, print, signature,  
      where = topenv(parent.frame()), edit = FALSE)  
untrace(what, signature = NULL, where = topenv(parent.frame()))
```

```
tracingState(on = NULL)  
.doTrace(expr, msg)  
returnValue(default = NULL)
```



```
what      quote()untracewhere = *
tracer    at
exit      function
at        tracer
print     TRUE
signature what
edit      editTRUE()edit
where     trace
           where

on        tracingStateTRUEFALSEFALSE
exprmsg   .doTraceexprbrowser()msg
default   returnValuedefault
```

```
tracesignaturewhat
"function"untrace

```

```
traceNULL
untrace
tracingState
on.exitreturnValue
```

```
trace()debugbrowser
```

```
tracesearch
```

browserrecoverquotesubstitute

```
require(stats)

## Very simple use
trace(sum)
hist(rnorm(100)) # shows about 3-4 calls to sum()
untrace(sum)

## Show how pt() is called from inside power.t.test():
if(FALSE)
  trace(pt) ## would show ~20 calls, but we want to see more:
  trace(pt, tracer = quote(cat(sprintf("tracing pt(*, ncp = %.15g)\n", ncp))),
        print = FALSE) # <- not showing typical extra
  power.t.test(20, 1, power=0.8, sd=NULL) ##--> showing the ncp root finding:
  untrace(pt)

f <- function(x, y) {
  y <- pmax(y, 0.001)
  if (x > 0) x ^ y else stop("x must be positive")
}

## arrange to call the browser on entering and exiting
## function f
trace("f", quote(browser(skipCalls = 4)),
      exit = quote(browser(skipCalls = 4)))

## instead, conditionally assign some data, and then browse
## on exit, but only then. Don't bother me otherwise

trace("f", quote(if(any(y < 0)) yOrig <- y),
      exit = quote(if(exists("yOrig")) browser(skipCalls = 4)),
      print = FALSE)

## Enter the browser just before stop() is called. First, find
## the step numbers

untrace(f) # (as it has changed f's body !)
as.list(body(f))
as.list(body(f)[[3]]) # -> stop(..) is [[4]]

## Now call the browser there

trace("f", quote(browser(skipCalls = 4)), at = list(c(3,4)))
## Not run:
f(-1,2) # --> enters browser just before stop(..)

## End(Not run)

## trace a utility function, with recover so we
## can browse in the calling functions as well.
```

```

trace("as.matrix", recover)

## turn off the tracing (that happened above)

untrace(c("f", "as.matrix"))

## Not run:
## Useful to find how system2() is called in a higher-up function:
trace(base::system2, quote(print(ls.str()))))

## End(Not run)

##----- Tracing hidden functions : need 'where = *'
##
## 'where' can be a function whose environment is meant:
trace(quote(ar.yw.default), where = ar)
a <- ar(rnorm(100)) # "Tracing ..."
untrace(quote(ar.yw.default), where = ar)

## trace() more than one function simultaneously:
##      expression(E1, E2, ...) here is equivalent to
##      c(quote(E1), quote(E2), quote(.*), ..)
trace(expression(ar.yw, ar.yw.default), where = ar)
a <- ar(rnorm(100)) # --> 2 x "Tracing ..."
# and turn it off:
untrace(expression(ar.yw, ar.yw.default), where = ar)

## Not run:
## trace calls to the function lm() that come from
## the nlme package.
trace("lm", where = asNamespace("nlme"))
  lm      (len ~ log(dose) * supp, ToothGrowth) -> fit1 # NOT traced
nlme::lmList(len ~ log(dose) | supp, ToothGrowth) -> fit2 # traced
untrace("lm", where = asNamespace("nlme"))

## End(Not run)

```

traceback

```

traceback()

.traceback()traceback(x, *).traceback(x)

      traceback(x = NULL, max.lines = getOption("traceback.max.lines",
                                                getOption("deparse.max.lines", -1L)))
.traceback(x = NULL, max.lines = getOption("traceback.max.lines",
                                                getOption("deparse.max.lines", -1L)))

x          NULL.Traceback
max.lines  xNULLlistpairlist

```

```

call.Tracebacktraceback.Call
trytryCatch
xoptions(error = function() traceback(3))traceback().traceback()
x
.traceback()traceback()callmax.lines

.traceback()max.linesmax.lines"truncated"
traceback().traceback()

.Traceback.Tracebackcall

```

```

foo <- function(x) { print(1); bar(2) }
bar <- function(x) { x + a.variable.which.does.not.exist }
## Not run:
foo(2) # gives a strange error
traceback()
## End(Not run)
## 2: bar(2)
## 1: foo(2)
bar
## Ah, this is the culprit ...

## This will print the stack trace at the time of the error.
options(error = function() traceback(3))

```

```

tracemem

```

```

tracemem(x)
untracemem(x)
retracemem(x, previous = NULL)

```

```

x                NULL
previous         tracememretracemem

```

tracememuntracememretracemem

duplicate.C.Fortran
tracememretracemem()
tracingState
traceNULL

NULL

capabilities("profmem")
traceRprofmem
<https://developer.r-project.org/memory-profiling.html>

```
## Not run:
a <- 1:10
tracemem(a)
## b and a share memory
b <- a
b[1] <- 1
untracemem(a)

## copying in lm: less than R <= 2.15.0
d <- stats::rnorm(10)
tracemem(d)
lm(d ~ a+log(b))

## f is not a copy and is not traced
f <- d[-1]
f+1
## indicate that f should be traced as a copy of d
retracemem(f, retracemem(d))
f+1

## End(Not run)
```

transform

transformtransform.defaulttransform.data.frame

transform(`_data`, ...)

```
_data
...          tag=value

...transform.data.frame_datanames(_data)_data_data
```

```
_data
```

```
transform
```

```
withinsubsetlistdata.frame
```

```
transform(airquality, Ozone = -Ozone)
transform(airquality, new = -Ozone, Temp = (Temp-32)/1.8)

attach(airquality)
transform(Ozone, logOzone = log(Ozone)) # marginally interesting ...
detach(airquality)
```

Trig

```
cospi(x)sinpi(x)tanpi(x)cos(pi*x)sin(pi*x)tan(pi*x)
```

```
cos(x)
sin(x)
tan(x)
```

```
acos(x)
asin(x)
atan(x)
atan2(y, x)
```

```
cospi(x)
sinpi(x)
tanpi(x)
```

xy

$\text{atan2}(y, x)$ (x, y) $\text{atan2}(y, x) == \text{atan}(y/x)$

$\pi/2$ cospi

$\text{cospi}(x)$ $\text{sinpi}(x)$ $\text{tanpi}(x)$ x

atan2 [Math](#)

ccospisinpitani ccospisinpitani

$\text{tanpi}(0.5)$ [NaN](#) 0.5

$\text{asinacos}(-\infty, -1]$ $[1, \infty)$

$\text{atan}(-\infty i, -1i]$ $[1i, \infty i)$

ccospisinpitani

atan2 [Math](#)

ccospisinpitani <https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1950.pdf>

```
x <- seq(-3, 7, by = 1/8)
tx <- cbind(x, cos(pi*x), cospi(x), sin(pi*x), sinpi(x),
            tan(pi*x), tanpi(x), deparse.level=2)
op <- options(digits = 4, width = 90) # for nice formatting
head(tx)
tx[ (x %% 1) %in% c(0, 0.5) ,]
options(op)
```

trimws

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \\t\\r\\n]")
```

```
x
which      "left" "right"
whitespace
```

```
sub(re, "", *, perl = TRUE)[ \\t\\r\\n][\\h\\v]https://www.pcre.org
```

```
x <- " Some text. "
x
trimws(x)
trimws(x, "l")
trimws(x, "r")

## Unicode --> need "stronger" 'whitespace' to match all :
tt <- "text with unicode 'non breakable space'."
xu <- paste(" \\t\\v", tt, "\\u00a0 \\n\\r")
(tu <- trimws(xu, whitespace = "[\\h\\v]"))
stopifnot(identical(tu, tt))
```

try

```
try
```

```
try(expr, silent = FALSE,
     outFile = getOption("try.outFile", default = stderr()))
```

```
expr
silent
outFile      cat(*, file = outFile)silent
```



```
try(stderroptions("show.error.messages")silent = TRUEgeterrmessage
trytryCatchtry(expr, silent = TRUE)tryCatch(expr, error = function(e) e)
outFilestdout()
```

```
options(try.outFile = stdout())
```

```
stderr()try()Sweave
```

```
expr"try-error""condition"
```

```
if (class(res) == "try-error"))
```

```
if(inherits(res, "try-error"))
```

```
optionsgeterrmessagetryCatch
```

```
assertCondition
```

```
## this example will not work correctly in example(try), but
## it does work correctly if pasted in
options(show.error.messages = FALSE)
try(log("a"))
print(.Last.value)
options(show.error.messages = TRUE)
```

```
## alternatively,
print(try(log("a"), TRUE))
```

```
## run a simulation, keep only the results that worked.
set.seed(123)
x <- stats::rnorm(50)
doit <- function(x)
{
  x <- sample(x, replace = TRUE)
  if(length(unique(x)) > 30) mean(x)
  else stop("too few unique points")
}
## alternative 1
res <- lapply(1:100, function(i) try(doit(x), TRUE))
## alternative 2
## Not run: res <- vector("list", 100)
for(i in 1:100) res[[i]] <- try(doit(x), TRUE)
## End(Not run)
unlist(res[sapply(res, function(x) !inherits(x, "try-error"))])
```

typeof

typeof

typeof(x)

x

TypeTablesrc/main/util.c"logical""integer""double""complex""character""raw"
"list""NULL""closure""special""builtin""environment""S4""symbol""pairlist"
"promise""object""language""char""..."any""expression""externalptr""bytecode"
"weakref"

[modestorage.mode](#)

[isS4](#)

typeof(2)

mode(2)

for a table of examples, see ?mode / examples(mode)

unique

uniquex

unique(x, incomparables = FALSE, ...)

Default S3 method:

unique(x, incomparables = FALSE, fromLast = FALSE,
 nmax = NA, ...)

S3 method for class 'matrix'

unique(x, incomparables = FALSE, MARGIN = 1,
 fromLast = FALSE, ...)

S3 method for class 'array'

unique(x, incomparables = FALSE, MARGIN = 1,
 fromLast = FALSE, ...)

```

x          is.vectorNULL
incomparables FALSEx
fromLast   namesdimnames
nmax       duplicated
...
MARGIN

```

```

is.vector
MARGINMARGIN = 2
uniqrle
"NA"NaNmatchduplicated
incomparables

```

```

x"POSIXct""Date""difftime"

```

```

[, drop = FALSE]x

```

```

vector $O(n^2)$ 

```

```

duplicated
rleuniq -c

```

```

x <- c(3:5, 11:8, 8 + 0:5)
(ux <- unique(x))
(u2 <- unique(x, fromLast = TRUE)) # different order
stopifnot(identical(sort(ux), sort(u2)))

length(unique(sample(100, 100, replace = TRUE)))
## approximately 100(1 - 1/e) = 63.21

unique(iris)

```

units

```
units(x)
units(x) <- value
```

x
value

"difftime"

unlink

unlinkx

```
unlink(x, recursive = FALSE, force = FALSE, expand = TRUE)
```

x
recursive
force
expand [path.expand](#)

```
recursive = FALSE
unlink(x, recursive = TRUE)
*?Sys.glob....~*?[a-z]manglob* ? [unlinkfile.remove
recursive = TRUE
```

```
01recursive = FALSEx
```

[file.remove](#)

unlist

xunlistx

unlist(x, recursive = TRUE, use.names = TRUE)

x
recursive TRUEFALSEx
use.names

unlist[relist](#)unlistrelistable
recursive = FALSEx
[x](#)[factor](#)
xunlist
unlistxuse.names = FALSE

[lm](#)unlist(x)xx

NULL

[cas.listrelist](#)

unlist(options())
unlist(options(), use.names = FALSE)

l.ex <- list(a = list(1:5, LETTERS[1:5]), b = "Z", c = NA)
unlist(l.ex, recursive = FALSE)
unlist(l.ex, recursive = TRUE)

l1 <- list(a = "a", b = 2, c = pi+2i)
unlist(l1) # a character vector
l2 <- list(a = "a", b = as.name("b"), c = pi+2i)
unlist(l2) # remains a list

l1 <- list(as.name("sinc"), quote(a + b), 1:10, letters, expression(1+x))
utils::str(l1)
for(x in l1)
 stopifnot(identical(x, unlist(x)))

unname

namesdimnames

[namesdimnames](#)

unname(obj, force = FALSE)

obj

force dimnames[data.frame](#)

obj[namesdimnames](#)

require(graphics); require(stats)

```
## Answering a question on R-help (14 Oct 1999):
col3 <- 750+ 100*rt(1500, df = 3)
breaks <- factor(cut(col3, breaks = 360+5*(0:155)))
z <- table(breaks)
z[1:5] # The names are larger than the data ...
barplot(unname(z), axes = FALSE)
```

use

use(package, include.only)

package

include.only

[library](#)attach.required = FALSEDependsDESCRIPTION
useImportFromNAMESPACE

UseMethod

UseMethodNextMethod

UseMethod(generic, object)

NextMethod(generic = NULL, object = NULL, ...)

generic UseMethod

object UseMethod

...

classis.object

"matrix""array"mode(x)c("integer", "numeric")c("double", "numeric").class2(x)x

UseMethod("fun")c("first", "second")fun.firstfun.secondfun.default

methods

UseMethodUseMethod

NextMethodNextMethodNextMethodgeneric

NextMethodUseMethodget("print.ts")(AirPassengers)

UseMethodNextMethodtopenv

UseMethodUseMethodUseMethodUseMethodUseMethodobject

NextMethod...

NextMethodUseMethodfun.default[.Internal

.Generic.Method.Class

.Generic

.MethodOps

.ClassNextMethod"previous".Class.Class.Class

.GenericCallEnv.GenericDefEnv

.ClassNextMethod.Class

`methodsclass.class2getS3methodis.object`

`userhooks`

```
getHook(hookName)
setHook(hookName, value,
        action = c("append", "prepend", "replace"))

packageEvent(pkgname,
             event = c("onLoad", "attach", "detach", "onUnload"))
```

```
hookName
pkgname
event
value      action = "replace"NULL
action
```

```
setHookpackageEvent
setHook(hookName, NULL, "replace")
library.onLoadsetHook()funname(pkgname, pkgpath)try
```

```
.onLoad
setLoadAction
```

```
"onLoad"
```

```
.onAttach
```

```
"attach"
```



```
"detach"  
  .Last.lib
```

```
"onUnload"  
  .onUnload
```

```
getHook"detach""onUnload"  
  .userHooksEnv
```

```
getHooksetHookpackageEvent
```

```
.Rprofile
```

```
librarydetachloadNamespace  
::  
plot.newpersp
```

```
setHook(packageEvent("grDevices", "onLoad"),  
  function(...) grDevices::ps.options(horizontal = FALSE))
```

```
utf8Conversion
```

```
utf8ToInt(x)  
intToUtf8(x, multiple = FALSE, allow_surrogate_pairs = FALSE)
```

```
x  
multiple  
allow_surrogate_pairs  
  multiple = FALSE
```

```
00x10FFFF  
intToUtf8NAallow_surrogate_pairs = TRUE
```

```
utf8ToInt
intToUtf8(0) # Encoding NA "UTF-8"
NANA
```

```
0x10FFFF
0xD8000xDFFFutf8ToIntintToUtf8
0xFFFE0xFFFFutf8ToInt
```

<https://www.rfc-editor.org/rfc/rfc3629>
<https://www.unicode.org/versions/corrigendum9.html>

```
## will only display in some locales and fonts
intToUtf8(0x03B2L) # Greek beta

utf8ToInt("bi\u00dfchen")
utf8ToInt("\xfa\x4\xbf\xbf\x9f")

## A valid UTF-16 surrogate pair (for U+10437)
x <- c(0xD801, 0xDC37)
intToUtf8(x)
intToUtf8(x, TRUE)
(xx <- intToUtf8(x, , TRUE)) # will only display in some locales and fonts
charToRaw(xx)

## An example of how surrogate pairs might occur
x <- "\U10437"
charToRaw(x)
foo <- tempfile()
writeLines(x, file(foo, encoding = "UTF-16LE"))
## next two are OS-specific, but are mandated by POSIX
system(paste("od -x", foo)) # 2-byte units, correct on little-endian platforms
system(paste("od -t x1", foo)) # single bytes as hex
y <- readBin(foo, "integer", 2, 2, FALSE, endian = "little")
sprintf("%X", y)
intToUtf8(y, , TRUE)
```

UTF8filepaths

Encoding

filefile.fifo pipe gz file bz file xz file unz file.exists dir.exists unlink file.info list.files
file gz file
file.path
path.expand

file file.access file.append file.copy file.create file.exists file.info file.link
file.remove file.rename file.sym link dir.create dir.exists normalize Path path.expand
pipe Sys.glob Sys.junction unlink gz file bz file xz file unz
gz file load read RDS read.dcftargz con file
list.files list.dirs system

<e7><U+00e7>

validUTF8

validUTF8(x)
validEnc(x)

x

grep
validUTF8Encoding intToUtf8
validEnc conv

xNA

```

x <-
  ## from example(text)
  c("Jetzt", "no", "chli", "z\xc3\xbcrit\xc3\xbc\xc3\xbctsch:",
    "(noch", "ein", "bi\xc3\x9fchen", "Z\xc3\xbc", "deutsch)",
    ## from a CRAN check log
    "\xfa\xba\xbf\xbf\x9f")
validUTF8(x)
validEnc(x) # depends on the locale
Encoding(x) <-"UTF-8"
validEnc(x) # typically the last, x[10], is invalid

## Maybe advantageous to declare it "unknown":
G <- x ; Encoding(G[!validEnc(G)]) <- "unknown"
try( substr(x, 1,1) ) # gives 'invalid multibyte string' error in a UTF-8 locale
try( substr(G, 1,1) ) # works in a UTF-8 locale
nchar(G) # fine, too
## but it is not "more valid" typically:
all.equal(validEnc(x),
          validEnc(G)) # typically TRUE

```

vector

`typeoflistexpression`

```

vectoris.null(attributes(.))
as.vectormodeis.atomicmode="any"
is.vector(x)TRUExmode="any"

```

```

vector(mode = "logical", length = 0)
as.vector(x, mode = "any")
is.vector(x, mode = "any")

```

```

mode          "list""expression"vector"any"is.vector()typeofmode"any"
              is.vector(x, mode)typeof(x) == mode
length        length > .Machine$integer.max"double"
x

```

```

"logical""integer""numeric""double""complex""character""raw"
mode = "any"is.vectorTRUElistexpressionmodeFALSExas.vector
mode = "any"xas.vector(x)attributesnames
x"list""expression"as.vector(x)xas.vectorclass(x)
is.vectorFALSEas.vectormode = "any"

```

```

vectorFALSE0""nulNULL

as.vectortype

is.vectorTRUEFALSEis.vector(x, mode = "numeric")"integer""double"is.vector(x,
mode = "double")"double"

as.vector()

as.vector

mode"any""list""expression""symbol""pairlist""double""name"
mode = "any"

is.vector(as.vector(x, m), m)m"any"
m == "any"xlistexpressionnamesis.object

as.vectoris.vector"vector"as(x, "vector")is(x, "vector")
as.vector(x)is.vector(x)
mode"symbol""name""pairlist"as.nameas.pairlistmode

cis.numericis.list

df <- data.frame(x = 1:3, y = 5:7)
## Error:
try(as.vector(data.frame(x = 1:3, y = 5:7), mode = "numeric"))

x <- c(a = 1, b = 2)
is.vector(x)
as.vector(x)
all.equal(x, as.vector(x)) ## FALSE

###-- All the following are TRUE:
is.list(df)
! is.vector(df)
! is.vector(df, mode = "list")

is.vector(list(), mode = "list")

```

Vectorize

VectorizeFUN

```
Vectorize(FUN, vectorize.args = arg.names, SIMPLIFY = TRUE,  
          USE.NAMES = TRUE)
```

```
FUN          match.fun  
vectorize.args FUN  
SIMPLIFY     simplifyapply  
USE.NAMES
```

```
vectorize.argsVectorize...maply  
Vectorizeformals  
FUNvectorize.argsSIMPLIFYUSE.NAMESVectorizecombn
```

FUNmaply

```
# We use rep.int as rep is primitive  
vrep <- Vectorize(rep.int)  
vrep(1:4, 4:1)  
vrep(times = 1:4, x = 4:1)  
  
vrep <- Vectorize(rep.int, "times")  
vrep(times = 1:4, x = 4:1)  
  
f <- function(x = 1:3, y) c(x, y)  
vf <- Vectorize(f, SIMPLIFY = FALSE)  
f(1:3, 1:3)  
vf(1:3, 1:3)  
vf(y = 1:3) # Only vectorizes y, not x  
  
# Nonlinear regression contour plot, based on nls() example  
require(graphics)  
SS <- function(Vm, K, resp, conc) {  
  pred <- (Vm * conc)/(K + conc)  
  sum((resp - pred)^2 / pred)  
}  
vSS <- Vectorize(SS, c("Vm", "K"))  
Treated <- subset(Puromycin, state == "treated")  
  
Vm <- seq(140, 310, length.out = 50)  
K <- seq(0, 0.15, length.out = 40)  
SSvals <- outer(Vm, K, vSS, Treated$rate, Treated$conc)
```

```

contour(Vm, K, SSvals, levels = (1:10)^2, xlab = "Vm", ylab = "K")

# combn() has an argument named FUN
combnV <- Vectorize(function(x, m, FUNV = NULL) combn(x, m, FUN = FUNV),
                    vectorize.args = c("x", "m"))

combnV(4, 1:4)
combnV(4, 1:4, sum)

```

warning

```

warning(..., call. = TRUE, immediate. = FALSE, noBreaks. = FALSE,
        domain = NULL)
suppressWarnings(expr, classes = "warning")

```

```

...
call.
immediate.      getOption\("warn"\) <= 0
noBreaks.       options\(warn = 1\)
expr
domain          gettextNAstop
classes

```

```

options\("warn"\)
options\(warn = 1\)
warningSignalConditionwarn = getOption\("warn"\)warnwarning(immediate. = TRUE)warn
<= 0warn = 1
warnlast.warningwarnings
getOption\("warning.length"\)[... truncated]
muffleWarninginvokeRestartwarning
warning
suppressWarnings

```

[character](#)

```

stopmessagewarningsoptionswarn=
gettext

```

```

testit <- function() warning("testit")
testit() ## shows call
testit <- function() warning("problem in testit", call. = FALSE)
testit() ## no call
suppressWarnings(warning("testit"))

```

warnings

```
warningsprintlast.warning
```

```
warnings(...)
```

```
## S3 method for class 'warnings'
summary(object, ...)
```

```
## S3 method for class 'warnings'
print(x, tags,
      header = ngettext(n, "Warning message:\n", "Warning messages:\n"),
      ...)
## S3 method for class 'summary.warnings'
print(x, ...)
```

```

...          catwarnings()
object       "warnings"warnings()
x            "warnings""summary.warnings"
tags         missingcharacterlengthxpaste0(seq_len(n), ":", ")",n ≥ 2n <-
            length(x)
header       cat()

```

```

options("warn")last.warningwarnings()options(warn = 0)warning
length(last.warning)getOption("nwarnings")50

```

```
options(nwarnings = 10000)
```

```
last.warningoptions(warn)
```

```

warnings()"warnings"listNULL
summary(<warnings>)"summary.warnings"listunique(object)"counts"

```

```
last.warning
```


warning

```
## NB this example is intended to be pasted in,
##   rather than run by example()
ow <- options("warn")
for(w in -1:1) {
  options(warn = w); cat("\n warn =", w, "\n")
  for(i in 1:3) { cat(i,"..\n"); m <- matrix(1:7, 3,4) }
  cat("-----\n")
}
## at the end prints all three warnings, from the 'option(warn = 0)' above
options(ow) # reset to previous, typically 'warn = 0'
tail(warnings(), 2) # see the last two warnings only (via '[' method)

## Often the most useful way to look at many warnings:
summary(warnings())

op <- options(nwarnings = 10000) ## <- get "full statistics"
x <- 1:36; for(n in 1:13) for(m in 1:12) A <- matrix(x, n,m) # There were 105 warnings ...
summary(warnings())
options(op) # revert to previous (keeping 50 messages by default)
```

weekdays

```
weekdays(x, abbreviate)
## S3 method for class 'POSIXt'
weekdays(x, abbreviate = FALSE)
## S3 method for class 'Date'
weekdays(x, abbreviate = FALSE)
```

```
months(x, abbreviate)
## S3 method for class 'POSIXt'
months(x, abbreviate = FALSE)
## S3 method for class 'Date'
months(x, abbreviate = FALSE)
```

```
quarters(x, abbreviate)
## S3 method for class 'POSIXt'
quarters(x, ...)
## S3 method for class 'Date'
quarters(x, ...)
```

```

julian(x, ...)
## S3 method for class 'POSIXt'
julian(x, origin = as.POSIXct("1970-01-01", tz = "GMT"), ...)
## S3 method for class 'Date'
julian(x, origin = as.Date("1970-01-01"), ...)

```

```

x                "POSIXt""Date"
abbreviate
origin           "POSIXt""Date"
...

```

```

weekdaysmonthsSys.getlocale("LC_TIME")
quarters"Q1""Q4"
julian"origin"

```

`as.POSIXltstrftime`

`DateTimeClassesDateSys.getlocale("LC_TIME")months()weekdays()`

```

## first two are locale dependent:
weekdays(.leap.seconds)
months (.leap.seconds)
quarters(.leap.seconds)

## Show how easily you get month, day, year, day (of {month, week, yr}), ... :
## (remember to count from 0 (!): mon = 0..11, wday = 0..6, etc !!)

##' Transform (Time-)Date vector to convenient data frame :
dt2df <- function(dt, dName = deparse(substitute(dt))) {
  DF <- as.data.frame(unclass(as.POSIXlt( dt )))
  `names<-`(cbind(dt, DF, deparse.level=0L), c(dName, names(DF)))
}
## e.g.,
dt2df(.leap.seconds) # date+time
dt2df(Sys.Date() + 0:9) # date

##' Even simpler: Date -> Matrix - dropping time info {sec,min,hour, isdst}
d2mat <- function(x) simplify2array(unclass(as.POSIXlt(x))[4:7])
## e.g.,
d2mat(seq(as.Date("2000-02-02"), by=1, length.out=30)) # has R 1.0.0's release date

## Julian Day Number (JDN, https://en.wikipedia.org/wiki/Julian\_day)
## is the number of days since noon UTC on the first day of 4317 BCE.
## in the proleptic Julian calendar. To more recently, in
## 'Terrestrial Time' which differs from UTC by a few seconds
## See https://en.wikipedia.org/wiki/Terrestrial\_Time
julian(Sys.Date(), -2440588) # from a day
floor(as.numeric(julian(Sys.time())) + 2440587.5) # from a date-time

```

which

TRUE

```
which(x, arr.ind = FALSE, useNames = TRUE)
arrayInd(ind, .dim, .dimnames = NULL, useNames = FALSE)
```

```
x          logicalNAFALSE
arr.ind     x
ind         which(x)
.dim        dim(.)
.dimnames   dimnames(.)useNamesarrayInd()which(*,      arr.ind=TRUE)
            names(.dimnames).dimnames[[1]]
useNames    arrayInd()
```

```
arr.ind == FALSExlengthsum(x)TRUEx
(1:length(x))[x]xNAwhich(x)seq_along(x)[!is.na(x) & x]namesx
arr.ind == TRUExarraydimarrayInd(which(x), dim(x), dimnames(x))x
```

xtypeof

arr.ind

Logicwhich.minmatchamatch(a, x)min(which(x == a))

```
which(LETTERS == "R")
which(l1 <- c(TRUE, FALSE, TRUE, NA, FALSE, FALSE, TRUE)) #> 1 3 7
names(l1) <- letters[seq(l1)]
which(l1)
which((1:12)%2 == 0) # which are even?
which(1:10 > 3, arr.ind = TRUE)

( m <- matrix(1:12, 3, 4) )
div.3 <- m %% 3 == 0
which(div.3)
which(div.3, arr.ind = TRUE)
rownames(m) <- paste("Case", 1:3, sep = "_")
which(m %% 5 == 0, arr.ind = TRUE)

dim(m) <- c(2, 2, 3); m
```

```

which(div.3, arr.ind = FALSE)
which(div.3, arr.ind = TRUE)

vm <- c(m)
dim(vm) <- length(vm) #-- funny thing with length(dim(...)) == 1
which(div.3, arr.ind = TRUE)

```

which.min

```

which.min(x)
which.max(x)

```

```

x                                doubleminmax

```

NaN

```

integerlength(x) =: n ≥ 231doublexNAx

```

```

which(x == min(x, na.rm = TRUE))which(x == max(x, na.rm = TRUE))

```

xTRUEFALSE

```

logicalxFALSETRUEwhich.min(x)which.max(x)FALSETRUEFALSE < TRUEmatch(FALSE, x)
match(TRUE, x)

```

```

whichmax.colmax

```

```

arrayInd()

```

```

which.is.max

```

```

x <- c(1:4, 0:5, 11)
which.min(x)
which.max(x)

```

```

## it *does* work with NA's present, by discarding them:

```

```

presidents[1:30]
range(presidents, na.rm = TRUE)
which.min(presidents) # 28
which.max(presidents) # 2

```

```

## Find the first occurrence, i.e. the first TRUE, if there is at least one:

```

```

x <- rpois(10000, lambda = 10); x[sample.int(50, 20)] <- NA
## where is the first value >= 20 ?
which.max(x >= 20)

## Also works for lists (which can be coerced to numeric vectors):
which.min(list(A = 7, pi = pi)) ## -> c(pi = 2L)

```

with

```

with(data, expr, ...)
within(data, expr, ...)
## S3 method for class 'list'
within(data, expr, keepAttrs = TRUE, ...)

```

```

data      withsys.callwithin
expr      within()
          {
            a <- somefun()
            b <- otherfun()
            .....
            rm(unused1, temp)
          }
keepAttrs listwithin()logicalattributesdatanameskeepAttrs = FALSE
...

```

```

withexprdatadata
expr
withinexprdatawithintransform

```

```

withexprwithin

```

```

with()data
dataformulaatawith(data, ...)

```

<https://developer.r-project.org/nonstandard-eval.pdf>

[evalqattachassigntransform](#)

```

with(mtcars, mpg[cyl == 8 & disp > 350])
  # is the same as, but nicer than
mtcars$mpg[mtcars$cyl == 8 & mtcars$disp > 350]

require(stats); require(graphics)

# examples from glm:
with(data.frame(u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(118,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12)),
  list(summary(glm(lot1 ~ log(u), family = Gamma)),
    summary(glm(lot2 ~ log(u), family = Gamma))))

aq <- within(airquality, {      # Notice that multiple vars can be changed
  lOzone <- log(Ozone)
  Month <- factor(month.abb[Month])
  cTemp <- round((Temp - 32) * 5/9, 1) # From Fahrenheit to Celsius
  S.cT <- Solar.R / cTemp # using the newly created variable
  rm(Day, Temp)
})
head(aq)

# example from boxplot:
with(ToothGrowth, {
  boxplot(len ~ dose, boxwex = 0.25, at = 1:3 - 0.2,
    subset = (supp == "VC"), col = "yellow",
    main = "Guinea Pigs' Tooth Growth",
    xlab = "Vitamin C dose mg",
    ylab = "tooth length", ylim = c(0, 35))
  boxplot(len ~ dose, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
    subset = supp == "OJ", col = "orange")
  legend(2, 9, c("Ascorbic acid", "Orange juice"),
    fill = c("yellow", "orange"))
})

# alternate form that avoids subset argument:
with(subset(ToothGrowth, supp == "VC"),
  boxplot(len ~ dose, boxwex = 0.25, at = 1:3 - 0.2,
    col = "yellow", main = "Guinea Pigs' Tooth Growth",
    xlab = "Vitamin C dose mg",
    ylab = "tooth length", ylim = c(0, 35)))
with(subset(ToothGrowth, supp == "OJ"),
  boxplot(len ~ dose, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
    col = "orange"))
legend(2, 9, c("Ascorbic acid", "Orange juice"),
  fill = c("yellow", "orange"))

```

withVisible

```
withVisible(x)
```

```
x
```

```
expressioncall
```

```
value      x
visible
```

```
invisibleevalwithAutoprint()source()withVisible()
```

```
x <- 1
withVisible(x <- 1) # *$visible is FALSE
x
withVisible(x)      # *$visible is TRUE

# Wrap the call in evalq() for special handling

df <- data.frame(a = 1:5, b = 1:5)
evalq(withVisible(a + b), envir = df)
```

```
write
```

```
xconnection
cat()print()xncolumnsfile
numericcharacter"factor""Date""POSIXt"format
```

```
write(x, file = "data",
      ncolumns = if(is.character(x)) 1 else 5,
      append = FALSE, sep = " ")
```

```
x
file      connection""""stdout()
          .Platform$OS.type != "windows""|cmd"cmd
ncolumns
append    TRUEx
sep       sep = "\t"" "
```

```

writecat
write.tablewriteLinesscan
saveRDSsave

# Demonstrate default ncolumns, writing to the console
write(month.abb, "") # 1 element per line for "character"
write(stack.loss, "") # 5 elements per line for "numeric"

# Build a file with sequential calls
fil <- tempfile("data")
write("# Model settings", fil)
write(month.abb, fil, ncolumns = 6, append = TRUE)
write("\n# Initial parameter values", fil, append = TRUE)
write(sqrt(stack.loss), fil, append = TRUE)
if(interactive()) file.show(fil)
unlink(fil) # tidy up

```

writeLines

```

writeLines(text, con = stdout(), sep = "\n", useBytes = FALSE)

```

```

text
con
sep
useBytes

```

```

confilefile
"wt"
writeLinessepwriteChar
useBytesuseBytes = TRUEiconv"bytes"

```

```

connectionswriteCharwriteBinreadLinescat

```

xtfrm

x

xtfrm(x)

x

```
rankrank(x, ties.method = "min", na.last = "keep")NANA
factor
is.numeric(x)==>x[i]iis.nax
```

x

ranksortorder

zapsmall

zapsmalldigitsdrround(x, digits = dr)0

```
zapsmall(x, digits = getOption("digits"),
  mFUN = function(x, ina) max(abs(x[!ina])),
  min.d = 0L)
```

x roundlog10()

digits

mFUN function(x, ina)xlogicalina := is.na(x)abs(x)x

min.d round(x, digits=*)mFUN(*) > 0

```

x2 <- pi * 100^(-2:2)/10
  print( x2, digits = 4)
zapsmall( x2) # automatic digits
zapsmall( x2, digits = 4)
zapsmall(c(x2, Inf)) # round()s to integer ..
zapsmall(c(x2, Inf), min.d=-Inf) # everything is small wrt Inf

(z <- exp(1i*0:4*pi/2))
zapsmall(z)

zapShow <- function(x, ...) rbind(orig = x, zapped = zapsmall(x, ...))
zapShow(x2)

## using a *robust* mFUN
mF_rob <- function(x, ina) boxplot.stats(x, do.conf=FALSE)$stats[5]
## with robust mFUN(), 'Inf' is no longer distorting the picture:
zapShow(c(x2, Inf), mFUN = mF_rob)
zapShow(c(x2, Inf), mFUN = mF_rob, min.d = -5) # the same
zapShow(c(x2, 999), mFUN = mF_rob) # same *rounding* as w/ Inf
zapShow(c(x2, 999), mFUN = mF_rob, min.d = 3) # the same
zapShow(c(x2, 999), mFUN = mF_rob, min.d = 8) # small diff

```

zpackages

```
.packages
```

```
.packages(all.available = FALSE, lib.loc = NULL)
```

```
all.available TRUElib.loc
lib.loc      NULLNULL.libPaths()
```

```
.packages().packages(all.available = TRUE)lib.loc
find.packageinstalled.packages
```

```
all.available = TRUE
```

```
.packages(all.available = TRUE)find.packagerequire
```

```
all.available = TRUE.packages
```

```
library.libPathsinstalled.packages
```

```
(.packages())          # maybe just "base"  
.packages(all.available = TRUE) # return all available as character vector  
require(splines)  
(.packages())          # "splines", too  
detach("package:splines")
```

zutils

.standard_regexps()

.standard_regexpsvalid_package_namevalid_package_version

compiler

compile

```
cmpfun(f, options = NULL)
compile(e, env = .GlobalEnv, options = NULL, srcref = NULL)
cmpfile(infile, outfile, ascii = FALSE, env = .GlobalEnv,
         verbose = FALSE, options = NULL, version = NULL)
loadcmp(file, envir = .GlobalEnv, chdir = FALSE)
disassemble(code)
enableJIT(level)
compilePKGS(enable)
getCompilerOption(name, options)
setCompilerOptions(...)
```

```
f
options
env
srcref
fileinfileoutfile
                                outfileinfile.Rc
ascii
verbose
version                        NULL
envir
chdir
code
e
level                          03
enable                         TRUE
name
...
```

```

cmpfun
compileeval
cmpfileinfileoutfileinfile.Rc
loadcmpsys.source
disassemble
enableJITlevellevelleveloptimizeR_ENABLE_JITenableJIT3
compilePKGS_R_COMPILE_PKGS_R_COMPILE_PKGSINSTALL
cat
optionsoptimizesuppressAllsuppressUndefinedsuppressNoSuperAssignVaroptimize0
32suppressAllTRUEsuppressUndefinedTRUEsuppressNoSuperAssignVarTRUEsuppressAll
FALSEsuppressUndefinedTRUEsuppressNoSuperAssignVarTRUE
getCompilerOptionoptionsoptionssetCompilerOptionsetCompilerOptions(suppressAll
= TRUE, optimize = 3)
intgcc

```

```

oldJIT <- enableJIT(0)
# a simple example
f <- function(x) x+1
fc <- cmpfun(f)
fc(2)
disassemble(fc)

# old R version of lapply
la1 <- function(X, FUN, ...) {
  FUN <- match.fun(FUN)
  if (!is.list(X))
    X <- as.list(X)
  rval <- vector("list", length(X))
  for(i in seq_along(X))
    rval[i] <- list(FUN(X[[i]], ...))
  names(rval) <- names(X) # keep `names' !
  return(rval)
}
# a small variation
la2 <- function(X, FUN, ...) {
  FUN <- match.fun(FUN)
  if (!is.list(X))
    X <- as.list(X)
  rval <- vector("list", length(X))
  for(i in seq_along(X)) {
    v <- FUN(X[[i]], ...)
    if (is.null(v)) rval[i] <- list(v)
    else rval[[i]] <- v
  }
  names(rval) <- names(X) # keep `names' !
  return(rval)
}

```

```
}  
# Compiled versions  
la1c <- cmpfun(la1)  
la2c <- cmpfun(la2)  
# some timings  
x <- 1:10  
y <- 1:100  
  
system.time(for (i in 1:10000) lapply(x, is.null))  
system.time(for (i in 1:10000) la1(x, is.null))  
system.time(for (i in 1:10000) la1c(x, is.null))  
system.time(for (i in 1:10000) la2(x, is.null))  
system.time(for (i in 1:10000) la2c(x, is.null))  
system.time(for (i in 1:1000) lapply(y, is.null))  
system.time(for (i in 1:1000) la1(y, is.null))  
system.time(for (i in 1:1000) la1c(y, is.null))  
system.time(for (i in 1:1000) la2(y, is.null))  
system.time(for (i in 1:1000) la2c(y, is.null))  
  
enableJIT(oldJIT)
```


datasets

datasets-package

`library(help = "datasets")`

`<R-core@r-project.org>`

ability.cov

ability.cov

g

```
require(stats)
(ability.FA <- factanal(factors = 1, covmat = ability.cov))
update(ability.FA, factors = 2)
## The signs of factors and hence the signs of correlations are
## arbitrary with promax rotation.
update(ability.FA, factors = 2, rotation = "promax")
```

airmiles

airmiles

```
require(graphics)
plot(airmiles, main = "airmiles data",
     xlab = "Passenger-miles flown by U.S. commercial airlines", col = 4)
```

AirPassengers

AirPassengers

```
## The classic 'airline model', by full ML
(fit <- arima(log10(AirPassengers), c(0, 1, 1),
              seasonal = list(order = c(0, 1, 1), period = 12)))
update(fit, method = "CSS")
update(fit, x = window(log10(AirPassengers), start = 1954))
pred <- predict(fit, n.ahead = 24)
tl <- pred$pred - 1.96 * pred$se
tu <- pred$pred + 1.96 * pred$se
ts.plot(AirPassengers, 10^tl, 10^tu, log = "y", lty = c(1, 2, 2))

## full ML fit is the same if the series is reversed, CSS fit is not
ap0 <- rev(log10(AirPassengers))
attributes(ap0) <- attributes(AirPassengers)
arima(ap0, c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12))
arima(ap0, c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
      method = "CSS")

## Structural Time Series
ap <- log10(AirPassengers) - 2
(fit <- StructTS(ap, type = "BSM"))
par(mfrow = c(1, 2))
plot(cbind(ap, fitted(fit)), plot.type = "single")
plot(cbind(ap, tsSmooth(fit)), plot.type = "single")
```

airquality

airquality

```
[,1] Ozone  
[,2] Solar.R  
[,3] Wind  
[,4] Temp  
[,5] Month  
[,6] Day
```

```
Ozone  
Solar.R  
Wind  
Temp
```

```
require(graphics)  
pairs(airquality, panel = panel.smooth, main = "airquality data")
```

```
anscombe
```

```
xy
```

```
anscombe
```

```
x1x2x3  
      x4  
y1y2y3y4
```

```

require(stats); require(graphics)
summary(anscombe)

##-- now some "magic" to do the 4 regressions in a loop:
ff <- y ~ x
mods <- setNames(as.list(1:4), paste0("lm", 1:4))
for(i in 1:4) {
  ff[2:3] <- lapply(paste0(c("y","x"), i), as.name)
  ## or   ff[[2]] <- as.name(paste0("y", i))
  ##      ff[[3]] <- as.name(paste0("x", i))
  mods[[i]] <- lmi <- lm(ff, data = anscombe)
  print(anova(lmi))
}

## See how close they are (numerically!)
sapply(mods, coef)
lapply(mods, function(fm) coef(summary(fm)))

## Now, do what you should have done in the first place: PLOTS
op <- par(mfrow = c(2, 2), mar = 0.1+c(4,4,1,1), oma = c(0, 0, 2, 0))
for(i in 1:4) {
  ff[2:3] <- lapply(paste0(c("y","x"), i), as.name)
  plot(ff, data = anscombe, col = "red", pch = 21, bg = "orange", cex = 1.2,
        xlim = c(3, 19), ylim = c(3, 13))
  abline(mods[[i]], col = "blue")
}
mtext("Anscombe's 4 Regression data sets", outer = TRUE, cex = 1.5)
par(op)

```

attenu

attenu

event
mag
station
dist
accel

```
require(graphics)
## check the data class of the variables
sapply(attenu, data.class)
summary(attenu)
pairs(attenu, main = "attenu data")
coplot(accel ~ dist | as.factor(event), data = attenu, show.given = FALSE)
coplot(log(accel) ~ log(dist) | as.factor(event),
       data = attenu, panel = panel.smooth, show.given = FALSE)
```

attitude

attitude

```
require(stats); require(graphics)
pairs(attitude, main = "attitude data")
summary(attitude)
summary(fm1 <- lm(rating ~ ., data = attitude))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
           mar = c(4.1, 4.1, 2.1, 1.1))
plot(fm1)
summary(fm2 <- lm(rating ~ complaints, data = attitude))
plot(fm2)
par(opar)
```

austres

"ts"

austres

beavers

beaver1
beaver2

beaver1
beaver2

beaver1beaver2
0330

beaver1

```
require(graphics)
(yl <- range(beaver1$temp, beaver2$temp))
```

```
beaver.plot <- function(bdat, ...) {
  nam <- deparse(substitute(bdat))
  with(bdat, {
    # Hours since start of day:
    hours <- time %% 100 + 24*(day - day[1]) + (time %% 100)/60
    plot (hours, temp, type = "l", ...,
          main = paste(nam, "body temperature"))
  })
}
```

```

    abline(h = 37.5, col = "gray", lty = 2)
    is.act <- activ == 1
    points(hours[is.act], temp[is.act], col = 2, cex = .8)
  })
}
op <- par(mfrow = c(2, 1), mar = c(3, 3, 4, 2), mgp = 0.9 * 2:0)
  beaver.plot(beaver1, ylim = yl)
  beaver.plot(beaver2, ylim = yl)
par(op)

```

BJsales

BJsalesBJsales.lead"ts"

BJsales
BJsales.lead

<https://robjhyndman.com/TSDL/>

BOD

BOD

BOD

Time
demand

```

require(stats)
# simplest form of fitting a first-order model to these data
fm1 <- nls(demand ~ A*(1-exp(-exp(lrc)*Time)), data = BOD,
          start = c(A = 20, lrc = log(.35)))
coef(fm1)
fm1

# using the plinear algorithm (trace o/p differs by platform)
fm2 <- nls(demand ~ (1-exp(-exp(lrc)*Time)), data = BOD,
          start = c(lrc = log(.35)), algorithm = "plinear", trace = TRUE)

# using a self-starting model
fm3 <- nls(demand ~ SSasymptOrig(Time, A, lrc), data = BOD)
summary(fm3)

```

`cars`

`cars`

```

require(stats); require(graphics)
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1)
lines(lowess(cars$speed, cars$dist, f = 2/3, iter = 3), col = "red")
title(main = "cars data")
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, log = "xy")
title(main = "cars data (logarithmic scales)")
lines(lowess(cars$speed, cars$dist, f = 2/3, iter = 3), col = "red")
summary(fm1 <- lm(log(dist) ~ log(speed), data = cars))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
           mar = c(4.1, 4.1, 2.1, 1.1))
plot(fm1)

```



```

par(opar)

## An example of polynomial regression
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
      las = 1, xlim = c(0, 25))
d <- seq(0, 25, length.out = 200)
for(degree in 1:4) {
  fm <- lm(dist ~ poly(speed, degree), data = cars)
  assign(paste("cars", degree, sep = "."), fm)
  lines(d, predict(fm, data.frame(speed = d)), col = degree)
}
anova(cars.1, cars.2, cars.3, cars.4)

```

ChickWeight

ChickWeight

ChickWeight

c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")

1848

[as.data.frame]plotprint

[SSlogis](#)

```

require(graphics)
coplot(weight ~ Time | Chick, data = ChickWeight,
        type = "b", show.given = FALSE)

```

chickwts

chickwts

weight
feed

```
require(stats); require(graphics)
boxplot(weight ~ feed, data = chickwts, col = "lightgray",
        varwidth = TRUE, notch = TRUE, main = "chickwt data",
        ylab = "Weight at six weeks (gm)")
anova(fm1 <- lm(weight ~ feed, data = chickwts))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
           mar = c(4.1, 4.1, 2.1, 1.1))
plot(fm1)
par(opar)
```

C02

C02

C02

c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")

Qn1Qn2Qn3Mc1

QuebecMississippi

nonchilledchilled

μ/m^2

CO_2CO_2

[as.data.frame]plotprint

```
require(stats); require(graphics)

coplot(uptake ~ conc | Plant, data = C02, show.given = FALSE, type = "b")
## fit the data for the first plant
fm1 <- nls(uptake ~ SSasym(conc, Asym, lrc, c0),
  data = C02, subset = Plant == "Qn1")
summary(fm1)
## fit each plant separately
fm1list <- list()
for (pp in levels(C02$Plant)) {
  fm1list[[pp]] <- nls(uptake ~ SSasym(conc, Asym, lrc, c0),
    data = C02, subset = Plant == pp)
}
## check the coefficients by plant
print(sapply(fm1list, coef), digits = 3)
```

co2

2

co2

https://scrippsco2.ucsd.edu/data/atmospheric_co2/

```
require(graphics)
plot(co2, ylab = expression("Atmospheric concentration of CO"[2]),
  las = 1)
title(main = "co2 data set")
```

crimtab

crimtab

```
tableinteger42 × 22sum(crimtab)
rownames"9.4""9.5"colnames"142.24""144.78"
```

$4'7''9/16 - -8''9/164'8''$

<https://pbil.univ-lyon1.fr/R/donnees/criminals1902.txt>

```
require(stats)
dim(crimtab)
utils::str(crimtab)
## for nicer printing:
local({cT <- crimtab
      colnames(cT) <- substring(colnames(cT), 2, 3)
      print(cT, zero.print = " ")
})

## Repeat Student's experiment:

# 1) Reconstitute 3000 raw data for heights in inches and rounded to
#    nearest integer as in Student's paper:

(heIn <- round(as.numeric(colnames(crimtab)) / 2.54))
d.hei <- data.frame(height = rep(heIn, colSums(crimtab)))

# 2) shuffle the data:

set.seed(1)
d.hei <- d.hei[sample(1:3000), , drop = FALSE]
```

```

# 3) Make 750 samples each of size 4:

d.hei$sample <- as.factor(rep(1:750, each = 4))

# 4) Compute the means and standard deviations (n) for the 750 samples:

h.mean <- with(d.hei, tapply(height, sample, FUN = mean))
h.sd <- with(d.hei, tapply(height, sample, FUN = sd)) * sqrt(3/4)

# 5) Compute the difference between the mean of each sample and
#     the mean of the population and then divide by the
#     standard deviation of the sample:

zobs <- (h.mean - mean(d.hei[, "height"])) / h.sd

# 6) Replace infinite values by +/- 6 as in Student's paper:

zobs[infZ <- is.infinite(zobs)] # none of them
zobs[infZ] <- 6 * sign(zobs[infZ])

# 7) Plot the distribution:

require(grDevices); require(graphics)
hist(x = zobs, probability = TRUE, xlab = "Student's z",
     col = grey(0.8), border = grey(0.5),
     main = "Distribution of Student's z score for 'crimtab' data")

```

discoveries

discoveries

```

require(graphics)
plot(discoveries, ylab = "Number of important discoveries",
     las = 1)
title(main = "discoveries data set")

```

DNase

DNase

DNase

```
c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")
```

103

[as.data.frame]plotprint

```
require(stats); require(graphics)

coplot(density ~ conc | Run, data = DNase,
       show.given = FALSE, type = "b")
coplot(density ~ log(conc) | Run, data = DNase,
       show.given = FALSE, type = "b")
## fit a representative run
fm1 <- nls(density ~ SSlogis( log(conc), Asym, xmid, scal ),
          data = DNase, subset = Run == 1)
## compare with a four-parameter logistic
fm2 <- nls(density ~ SSfpl( log(conc), A, B, xmid, scal ),
          data = DNase, subset = Run == 1)
summary(fm2)
anova(fm1, fm2)
```

esoph

esoph

agegp

alcgp

tobgp

ncases
ncontrols

```
require(stats)
require(graphics) # for mosaicplot
summary(esoph)
## effects of alcohol, tobacco and interaction, age-adjusted
model1 <- glm(cbind(ncases, ncontrols) ~ agegp + tobgp * alcgp,
              data = esoph, family = binomial())
anova(model1)
## Try a linear effect of alcohol and tobacco
model2 <- glm(cbind(ncases, ncontrols) ~ agegp + unclass(tobgp)
              + unclass(alcgp),
              data = esoph, family = binomial())
summary(model2)
## Re-arrange data for a mosaic plot
ttt <- table(esoph$agegp, esoph$alcgp, esoph$tobgp)
o <- with(esoph, order(tobgp, alcgp, agegp))
ttt[ttt == 1] <- esoph$ncases[o]
tt1 <- table(esoph$agegp, esoph$alcgp, esoph$tobgp)
tt1[tt1 == 1] <- esoph$ncontrols[o]
tt <- array(c(ttt, tt1), c(dim(ttt),2),
            c(dimnames(ttt), list(c("Cancer", "control")))))
mosaicplot(tt, main = "esoph data set", color = TRUE)
```

euro

```
euro
euro.cross
```

```
euroeuro.cross
```

```
euro
euro.crossouter(1 / euro, euro)
```

```
cbind(euro)
```

```
## These relations hold:
euro == signif(euro, 6) # [6 digit precision in Euro's definition]
all(euro.cross == outer(1/euro, euro))
```

```
## Convert 20 Euro to Belgian Franc
20 * euro["BEF"]
## Convert 20 Austrian Schilling to Euro
20 / euro["ATS"]
## Convert 20 Spanish Pesetas to Italian Lira
20 * euro.cross["ESP", "ITL"]
```

```
require(graphics)
dotchart(euro,
  main = "euro data: 1 Euro in currency unit")
dotchart(1/euro,
  main = "euro data: 1 currency unit in Euros")
dotchart(log(euro, 10),
  main = "euro data: log10(1 Euro in currency unit)")
```

eurodist

```
eurodist
UScitiesD
```

```
eurodist
UScitiesD
```

```
dist
```

EuStockMarkets

EuStockMarkets

"mts"

faithful

faithful

eruptions
waiting

faithful\$eruptions

geyser

```

require(stats); require(graphics)
f.tit <- "faithful data: Eruptions of Old Faithful"

ne60 <- round(e60 <- 60 * faithful$eruptions)
all.equal(e60, ne60) # relative diff. ~ 1/10000
table(zapsmall(abs(e60 - ne60))) # 0, 0.02 or 0.04
faithful$better.eruptions <- ne60 / 60
te <- table(ne60)
te[te >= 4] # (too) many multiples of 5 !
plot(names(te), te, type = "h", main = f.tit, xlab = "Eruption time (sec)")

plot(faithful[, -3], main = f.tit,
     xlab = "Eruption time (min)",
     ylab = "Waiting time to next eruption (min)")
lines(lowess(faithful$eruptions, faithful$waiting, f = 2/3, iter = 3),
      col = "red")

```

Formaldehyde

Formaldehyde

carb
optden

```

require(stats); require(graphics)
plot(optden ~ carb, data = Formaldehyde,
     xlab = "Carbohydrate (ml)", ylab = "Optical Density",
     main = "Formaldehyde data", col = 4, las = 1)
abline(fm1 <- lm(optden ~ carb, data = Formaldehyde))
summary(fm1)
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0))
plot(fm1)
par(opar)

```

freeny

```
freeny  
freeny.x  
freeny.y
```

```
freeny.y  
freeny.xfreeny.y  
freenyylag.quarterly.revenueprice.indexincome.levelmarket.potential
```

```
require(stats); require(graphics)  
summary(freeny)  
pairs(freeny, main = "freeny data")  
# gives warning: freeny$y has class "ts"  
  
summary(fm1 <- lm(y ~ ., data = freeny))  
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),  
            mar = c(4.1, 4.1, 2.1, 1.1))  
plot(fm1)  
par(opar)
```

gait

```
gait
```

```
c(20, 39, 2)"Hip Angle""Knee Angle"
```

```
dimnames(gait)

Time seq(from = 0.025, to = 0.975, by = 0.05)
Subject "boy1""boy2""boy39"
Variable "Hip Angle""Knee Angle"
```

```
gaitboy19boy26
```

<https://CRAN.R-project.org/package=fda>

```
plot(gait[, 1, ], type = "b",
      xlim = range(gait[,1]), ylim = range(gait[,2]),
      xlab = "Hip Angle", ylab = "Knee Angle", main = "'gait' data : Boy 1")
mtext("all other boys", col = "thistle"); grid()
matlines(gait[, -1, 1], gait[, -1, 2], type = "l", lty = 1, col = adjustcolor("thistle", 1/3))

## The data array, two matrices :
op <- options(width = 128) # on a wide console
aperm(gait, c(2:1, 3))
options(op)
```

HairEyeColor

HairEyeColor

Hair
Eye
Sex

×Sex

<http://www.datavis.ca/sas/vcd/catdata/haireye.sas>

SexSex

<http://datavis.ca/papers/sugi/sugi17.pdf>

<http://www.datavis.ca/papers/asa92.html>

`chisq.testloglinmosaicplot`

```
require(graphics)
## Full mosaic
mosaicplot(HairEyeColor)
## Aggregate over sex (as in Snee's original data)
x <- apply(HairEyeColor, c(1, 2), sum)
x
mosaicplot(x, main = "Relation between hair and eye color")
```

Harman23.cor

Harman23.cor

```
require(stats)
(Harman23.FA <- factanal(factors = 1, covmat = Harman23.cor))
for(factors in 2:4) print(update(Harman23.FA, factors = factors))
```

Harman74.cor

Harman74.cor

```
require(stats)
(Harman74.FA <- factanal(factors = 1, covmat = Harman74.cor))
for(factors in 2:5) print(update(Harman74.FA, factors = factors))
Harman74.FA <- factanal(factors = 5, covmat = Harman74.cor,
                        rotation = "promax")
print(Harman74.FA$loadings, sort = TRUE)
```

Indometh

Indometh

Indometh

```
c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")
```

```
[as.data.frame]plotprint
```

SSbiexp

infert

infert

```
require(stats)
model1 <- glm(case ~ spontaneous+induced, data = infert, family = binomial())
summary(model1)
## adjusted for other potential confounders:
summary(model2 <- glm(case ~ age+parity+education+spontaneous+induced,
                      data = infert, family = binomial()))
## Really should be analysed by conditional logistic regression
## which is in the survival package
if(require(survival)){
  model3 <- clogit(case ~ spontaneous+induced+strata(stratum), data = infert)
  print(summary(model3))
  detach() # survival (conflicts)
}
```

InsectSprays

InsectSprays

count
spray

```
require(stats); require(graphics)
boxplot(count ~ spray, data = InsectSprays,
        xlab = "Type of spray", ylab = "Insect count",
        main = "InsectSprays data", varwidth = TRUE, col = "lightgray")
fm1 <- aov(count ~ spray, data = InsectSprays)
summary(fm1)
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0))
plot(fm1)
fm2 <- aov(sqrt(count) ~ spray, data = InsectSprays)
summary(fm2)
plot(fm2)
par(opar)
```

iris

iris
iris3

irisSepal.LengthSepal.WidthPetal.LengthPetal.WidthSpecies
iris3Sepal L.Sepal W.Petal L.Petal W.


```
iris3iris
```

```
matplotiris
```

```
summary(iris)
```

```
## Fisher's (1936) research question: whether (compound measurements of)
## Iris versicolor "differs twice as much from I. setosa as from I. virginica"
pairs(iris[1:4], col = iris$Species)
legend(0.5, 1, levels(iris$Species), fill = 1:3, bty = "n",
       horiz = TRUE, xjust = 0.5, yjust = 0, xpd = TRUE)

## equivalence of legacy array (iris3) and data.frame (iris) representation
dni3 <- dimnames(iris3)
ii <- data.frame(matrix(aperm(iris3, c(1,3,2)), ncol = 4,
                          dimnames = list(NULL, sub(" L.", ".Length",
                                                    sub(" W.", ".Width", dni3[[2]]))),
                    Species = gl(3, 50, labels = sub("S", "s", sub("V", "v", dni3[[3]]))))
stopifnot(all.equal(ii, iris))
```

```
islands
```

```
islands
```

```
require(graphics)
dotchart(log(islands, 10),
         main = "islands data: log10(area) (log10(sq. miles))")
dotchart(log(islands[order(islands)], 10),
         main = "islands data: log10(area) (log10(sq. miles))")
```

JohnsonJohnson

JohnsonJohnson

```
require(stats); require(graphics)
JJ <- log10(JohnsonJohnson)
plot(JJ)
## This example gives a possible-non-convergence warning on some
## platforms, but does seem to converge on x86 Linux and Windows.
(fit <- StructTS(JJ, type = "BSM"))
tsdiag(fit)
sm <- tsSmooth(fit)
plot(cbind(JJ, sm[, 1], sm[, 3]-0.5), plot.type = "single",
      col = c("black", "green", "blue"))
abline(h = -0.5, col = "grey60")

monthplot(fit)
```

LakeHuron

LakeHuron

lh

lh

LifeCycleSavings

LifeCycleSavings

sr
pop15
pop75
dpi
ddpi

```
require(stats); require(graphics)
pairs(LifeCycleSavings, panel = panel.smooth,
      main = "LifeCycleSavings data")
fm1 <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
summary(fm1)
```

Loblolly

Loblolly

Loblolly

c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")

[as.data.frame]plotprint

```
require(stats); require(graphics)
plot(height ~ age, data = Loblolly, subset = Seed == 329,
      xlab = "Tree age (yr)", las = 1,
      ylab = "Tree height (ft)",
      main = "Loblolly data and fitted curve (Seed 329 only)")
fm1 <- nls(height ~ SSasymp(age, Asym, R0, lrc),
          data = Loblolly, subset = Seed == 329)
age <- seq(0, 30, length.out = 101)
lines(age, predict(fm1, list(age = age)))
```

longley

longley

$n = 16$

GNP.deflator 1954 = 100

GNP

Unemployed

Armed.Forces

Population \geq

Year

Employed

lm(Employed ~ .)

```
require(stats); require(graphics)
## give the data set in the form it was used in S-PLUS:
longley.x <- data.matrix(longley[, 1:6])
longley.y <- longley[, "Employed"]
pairs(longley, main = "longley data")
summary(fm1 <- lm(Employed ~ ., data = longley))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
            mar = c(4.1, 4.1, 2.1, 1.1))
plot(fm1)
par(opar)
```

lynx

lynx

morley

299000

morley

Expt
Run
Speed

richelson

```
require(stats); require(graphics)
richelson <- transform(morley,
                      Expt = factor(Expt), Run = factor(Run))
xtabs(~ Expt + Run, data = richelson) # 5 x 20 balanced (two-way)
plot(Speed ~ Expt, data = richelson,
     main = "Speed of Light Data", xlab = "Experiment No.")
fm <- aov(Speed ~ Run + Expt, data = richelson)
summary(fm)
fm0 <- update(fm, . ~ . - Run)
anova(fm0, fm)
```

mtcars

mtcars

mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb

```
require(graphics)
pairs(mtcars, main = "mtcars data", gap = 1/4)
coplot(mpg ~ disp | as.factor(cyl), data = mtcars,
       panel = panel.smooth, rows = 1)
## possibly more meaningful, e.g., for summary() or bivariate plots:
mtcars2 <- within(mtcars, {
  vs <- factor(vs, labels = c("V", "S"))
  am <- factor(am, labels = c("automatic", "manual"))
  cyl <- ordered(cyl)
  gear <- ordered(gear)
  carb <- ordered(carb)
})
summary(mtcars2)
```

nhtemp

nhtemp

```
require(stats); require(graphics)
plot(nhtemp, main = "nhtemp data",
     ylab = "Mean annual temperature in New Haven, CT (deg. F)")
```

Nile

Assuan $10^8 m^3$

Nile

```
require(stats); require(graphics)
par(mfrow = c(2, 2))
plot(Nile)
acf(Nile)
pacf(Nile)
ar(Nile) # selects order 2
cpgram(ar(Nile)$resid)
par(mfrow = c(1, 1))
arima(Nile, c(2, 0, 0))

## Now consider missing values, following Durbin & Koopman
NileNA <- Nile
NileNA[c(21:40, 61:80)] <- NA
arima(NileNA, c(2, 0, 0))
plot(NileNA)
pred <-
  predict(arima(window(NileNA, 1871, 1890), c(2, 0, 0)), n.ahead = 20)
lines(pred$pred, lty = 3, col = "red")
lines(pred$pred + 2*pred$se, lty = 2, col = "blue")
lines(pred$pred - 2*pred$se, lty = 2, col = "blue")
pred <-
  predict(arima(window(NileNA, 1871, 1930), c(2, 0, 0)), n.ahead = 20)
lines(pred$pred, lty = 3, col = "red")
lines(pred$pred + 2*pred$se, lty = 2, col = "blue")
lines(pred$pred - 2*pred$se, lty = 2, col = "blue")

## Structural time series models
par(mfrow = c(3, 1))
plot(Nile)
## local level model
(fit <- StructTS(Nile, type = "level"))
lines(fitted(fit), lty = 2) # contemporaneous smoothing
lines(tsSmooth(fit), lty = 2, col = 4) # fixed-interval smoothing
```



```

plot(residuals(fit)); abline(h = 0, lty = 3)
## local trend model
(fit2 <- StructTS(Nile, type = "trend")) ## constant trend fitted
pred <- predict(fit, n.ahead = 30)
## with 50% confidence interval
ts.plot(Nile, pred$pred,
        pred$pred + 0.67*pred$se, pred$pred -0.67*pred$se)

## Now consider missing values
plot(NileNA)
(fit3 <- StructTS(NileNA, type = "level"))
lines(fitted(fit3), lty = 2)
lines(tsSmooth(fit3), lty = 3)
plot(residuals(fit3)); abline(h = 0, lty = 3)

```

nottem

nottem

```

require(stats); require(graphics)
nott <- window(nottem, end = c(1936,12))
fit <- arima(nott, order = c(1,0,0), list(order = c(2,1,0), period = 12))
nott.fore <- predict(fit, n.ahead = 36)
ts.plot(nott, nott.fore$pred, nott.fore$pred+2*nott.fore$se,
        nott.fore$pred-2*nott.fore$se, gpars = list(col = c(1,1,4,4)))

```

npk

npk

npk

block

N

P

K

yield

```

options(contrasts = c("contr.sum", "contr.poly"))
npk.aov <- aov(yield ~ block + N*P*K, npk)
npk.aov
summary(npk.aov)
coef(npk.aov)
options(contrasts = c("contr.treatment", "contr.poly"))
npk.aov1 <- aov(yield ~ block + N + K, data = npk)
summary.lm(npk.aov1)
se.contrast(npk.aov1, list(N=="0", N=="1"), data = npk)
model.tables(npk.aov1, type = "means", se = TRUE)

```

occupationalStatus

occupationalStatus

[table](#)origin1:8destination1:8

```
require(stats); require(graphics)
```

```
plot(occupationalStatus)
```

```
## Fit a uniform association model separating diagonal effects
Diag <- as.factor(diag(1:8))
Rscore <- scale(as.numeric(row(occupationalStatus)), scale = FALSE)
Cscore <- scale(as.numeric(col(occupationalStatus)), scale = FALSE)
modUnif <- glm(Freq ~ origin + destination + Diag + Rscore:Cscore,
               family = poisson, data = occupationalStatus)
```

```
summary(modUnif)
plot(modUnif) # 4 plots, with warning about h_ii ~= 1
```

Orange

Orange

Orange

```
c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")
```

```
[as.data.frame]plotprint
```

```
require(stats); require(graphics)
coplot(circumference ~ age | Tree, data = Orange, show.given = FALSE)
fm1 <- nls(circumference ~ SSlogis(age, Asym, xmid, scal),
          data = Orange, subset = Tree == 3)
plot(circumference ~ age, data = Orange, subset = Tree == 3,
     xlab = "Tree age (days since 1968/12/31)",
     ylab = "Tree circumference (mm)", las = 1,
     main = "Orange tree data and fitted model (Tree 3 only)")
age <- seq(0, 1600, length.out = 101)
lines(age, predict(fm1, list(age = age)))
```

OrchardSprays

OrchardSprays

rowpos
colpos
treatment
decrease

8 × 8

```
require(graphics)  
pairs(OrchardSprays, main = "OrchardSprays data")
```

penguins

penguinspenguins_raw

penguins
penguins_raw

penguins

species **factor**AdelieChinstrapGentoo

island **factor**BiscoeDreamTorgersen

bill_len **numeric**

bill_dep **numeric**

flipper_len **integer**

body_mass **integer**

sex **factor**femalemale

year **integer**

penguins_rawpenguins

Species **character**

Island **character**

Culmen Length (mm) **numeric**

Culmen Depth (mm) **numeric**

Flipper Length (mm) **numeric**

Body Mass (g) **numeric**

Sex **character**

Date Egg **Date**yearpenguins

penguins_raw

studyName **character**

Sample Number **numeric**

Region **character**

Stage **character**

Individual ID **character**

Clutch Completion **character**

Delta 15 N (o/oo) **numeric**

Delta 13 C (o/oo) **numeric**

Comments **character**

iris

penguins

```

## view summaries
summary(penguins)
summary(penguins_raw) # not useful for character vectors
## convert character vectors to factors first
dFactor <- function(dat) {
  dat[] <- lapply(dat, \(.) if (is.character(.)) as.factor(.) else .)
  dat
}
summary(dFactor(penguins_raw))

## visualise distribution across factors
plot(island ~ species, data = penguins)
plot(sex ~ interaction(island, species, sep = "\n"), data = penguins)

## bill depth vs. length by species (color) and sex (symbol):
## positive correlations for all species, males tend to have bigger bills
sym <- c(1, 16)
pal <- c("darkorange", "purple", "cyan4")
plot(bill_dep ~ bill_len, data = penguins, pch = sym[sex], col = pal[species])

## simplified sex dimorphism analysis for Adelie species:
## proportion of males increases with several size measurements
adelie <- subset(penguins, species == "Adelie")
plot(sex ~ bill_len, data = adelie)
plot(sex ~ bill_dep, data = adelie)
plot(sex ~ body_mass, data = adelie)
m <- glm(sex ~ bill_len + bill_dep + body_mass, data = adelie, family = binomial)
summary(m)

## Produce the long variable names as from {palmerpenguins} pkg:
long_nms <- sub("len", "length_mm",
               sub("dep", "depth_mm",
                  sub("mass", "mass_g", colnames(penguins))))
## compare long and short names:
noquote(rbind(long_nms, nms = colnames(penguins)))

## Not run: # << keeping shorter 'penguins' names in this example:
colnames(penguins) <- long_nms

## End(Not run)

```

PlantGrowth

PlantGrowth

weight
group

group

```
## One factor ANOVA example from Dobson's book, cf. Table 7.4:  
require(stats); require(graphics)  
boxplot(weight ~ group, data = PlantGrowth, main = "PlantGrowth data",  
        ylab = "Dried weight of plants", col = "lightgray",  
        notch = TRUE, varwidth = TRUE)  
anova(lm(weight ~ group, data = PlantGrowth))
```

precip

precip

"Cincinnati"

```
require(graphics)
dotchart(precip[order(precip)], main = "precip data")
title(sub = "Average annual precipitation (in.)")

## Old ("wrong") version of dataset (just name change):
precip.0 <- local({
  p <- precip; names(p)[names(p) == "Cincinnati"] <- "Cincinnati" ; p })
stopifnot(all(precip == precip.0),
  match("Cincinnati", names(precip)) == 46,
  identical(names(precip)[-46], names(precip.0)[-46]))
```

presidents

presidents

```
require(stats); require(graphics)
plot(presidents, las = 1, ylab = "Approval rating (%)",
  main = "presidents data")
```

pressure

pressure

temperature
pressure

```
require(graphics)
plot(pressure, xlab = "Temperature (deg C)",
      ylab = "Pressure (mm of Hg)",
      main = "pressure data: Vapor Pressure of Mercury")
plot(pressure, xlab = "Temperature (deg C)", log = "y",
      ylab = "Pressure (mm of Hg)",
      main = "pressure data: Vapor Pressure of Mercury")
```

Puromycin

Puromycin

Puromycin

conc

rate

state treateduntreated

SSmicmen

```

require(stats); require(graphics)

plot(rate ~ conc, data = Puromycin, las = 1,
      xlab = "Substrate concentration (ppm)",
      ylab = "Reaction velocity (counts/min/min)",
      pch = as.integer(Puromycin$state),
      col = as.integer(Puromycin$state),
      main = "Puromycin data and fitted Michaelis-Menten curves")
## simplest form of fitting the Michaelis-Menten model to these data
fm1 <- nls(rate ~ Vm * conc/(K + conc), data = Puromycin,
           subset = state == "treated",
           start = c(Vm = 200, K = 0.05))
fm2 <- nls(rate ~ Vm * conc/(K + conc), data = Puromycin,
           subset = state == "untreated",
           start = c(Vm = 160, K = 0.05))
summary(fm1)
summary(fm2)
## add fitted lines to the plot
conc <- seq(0, 1.2, length.out = 101)
lines(conc, predict(fm1, list(conc = conc)), lty = 1, col = 1)
lines(conc, predict(fm2, list(conc = conc)), lty = 2, col = 2)
legend(0.8, 120, levels(Puromycin$state),
      col = 1:2, lty = 1:2, pch = 1:2)

## using partial linearity
fm3 <- nls(rate ~ conc/(K + conc), data = Puromycin,
           subset = state == "treated", start = c(K = 0.05),
           algorithm = "plinear")

```

quakes

quakes

```

lat
long
depth
mag
stations

```

```
require(graphics)
pairs(quakes, main = "Fiji Earthquakes, N = 1000", cex.main = 1.2, pch = ".")
```

randu

randu

xyz

```
## We could re-generate the dataset by the following R code
seed <- as.double(1)
RANDU <- function() {
  seed <- ((2^16 + 3) * seed) %% (2^31)
  seed/(2^31)
}
myrandu <- matrix(NA_real_, 400, 3, dimnames = list(NULL, c("x","y","z")))
for(i in 1:400) {
  U <- c(RANDU(), RANDU(), RANDU(), RANDU(), RANDU())
  myrandu[i,] <- round(U[1:3], 6)
}
stopifnot(all.equal(randu, as.data.frame(myrandu), tolerance = 1e-5))
```

rivers

rivers

rock

rock

area
peri
shape
perm

sleep

sleep

group

```
require(stats)
## Student's paired t-test
with(sleep,
      t.test(extra[group == 1],
              extra[group == 2], paired = TRUE))

## The sleep *prolongations*
sleep1 <- with(sleep, extra[group == 2] - extra[group == 1])
summary(sleep1)
stripchart(sleep1, method = "stack", xlab = "hours",
           main = "Sleep prolongation (n = 10)")
boxplot(sleep1, horizontal = TRUE, add = TRUE,
        at = .6, pars = list(boxwex = 0.5, staplewex = 0.25))
```

stackloss

stackloss

stack.x
stack.loss

stackloss

Air Flow
Water Temp
Acid Conc.
stack.loss

stack.xstack.loss

33

Air FlowWater TempAcid Conc.stack.loss

```
require(stats)
summary(lm.stack <- lm(stack.loss ~ stack.x))
```

state

```
state.abb  
state.area  
state.center  
state.division  
state.name  
state.region  
state.x77
```

```
state.abb  
state.area  
state.center xy  
state.division factor  
state.name  
state.region factor  
state.x77
```

```
Population  
Income  
Illiteracy  
Life Exp  
Murder  
HS Grad  
Frost  
Area
```

```
(cm(1760 * 3 * 12) / 100 / 1000)^2km^22.589988110336km^2
```

```

(dst <- dxy <- data.frame(state.center, row.names=state.abb))
## Alaska and Hawaii are placed just off the West Coast (for compact map drawing):
dst[c("AK", "HI"),]
## state.center2 := version of state.center with "correct" coordinates for AK & HI:
## From https://pubs.usgs.gov/gip/Elevations-Distances/elvadist.html#Geographic%20Centers
##   Alaska   63°50' N., 152°00' W., 60 miles northwest of Mount McKinley
##   Hawaii   20°15' N., 156°20' W., off Maui Island
dxy["AK",] <- c(-152. , 63.83) # or c(-152.11, 65.17)
dxy["HI",] <- c(-156.33, 20.25) # or c(-156.69, 20.89)
state.center2 <- as.list(dxy)

plot(dxy, asp=1.2, pch=3, col=2)
text(state.center2, state.abb, cex=1/2, pos=4, offset=1/4)
i <- c("AK", "HI")
do.call(arrows, c(setNames(c(dst[i,], dxy[i,]), c("x0", "y0", "x1", "y1")),
                    col=adjustcolor(4, .7), length=1/8))
points(dst[i,], col=2)
if(FALSE) { # if(require("maps")) {
  map("state", interior = FALSE, add = TRUE)
  map("state", boundary = FALSE, lty = 2, add = TRUE)
}

```

sunspot.month

sunspot.month
sunspot.m2014sunspotssunspot.month

sunspot.month
sunspot.m2014

sunspot.yearsunspot.m2014sunspot.month"ts"

<https://www.sidc.be/SILSO/datafiles>
sunspot.month

sunspot.monthsunspots

```

require(stats); require(graphics)
## Compare the monthly series
plot (sunspot.month,
      main="sunspot.month & sunspots [package 'datasets']", col=2)
lines(sunspots) # -> clear recalibration (to *larger* values)
at. <- seq(1750, 2030, by=10)
atyr <- at.[at. %% 50 != 0] ##
Axis(time(sunspot.month), at = atyr, side = 1,
      tck= -1/100, padj = -1.5, cex.axis = 3/4)

## Now look at the difference :
all(tsp(sunspots)      [c(1,3)] ==
    tsp(sunspot.month)[c(1,3)]) ## Start & Periodicity are the same
n1 <- length(sunspots)
table(eq <- sunspots == sunspot.m2014[1:n1]) #> 143 are different !
i <- which(!eq)
rug(time(eq)[i])
s1 <- sunspots[i] ; s2 <- sunspot.m2014[i]
cbind(i = i, time = time(sunspots)[i], sunspots = s1, ss.month = s2,
      perc.diff = round(100*2*abs(s1-s2)/(s1+s2), 1))

## How to recreate the "old" sunspot.month (R <= 3.0.3) =: sunspot.month.0
.sunspot.diff <- cbind(
  i = c(1202L, 1256L, 1258L, 1301L, 1407L, 1429L, 1452L, 1455L,
        1663L, 2151L, 2329L, 2498L, 2594L, 2694L, 2819L),
  res10 = c(1L, 1L, 1L, -1L, -1L, -1L, 1L, -1L,
            1L, 1L, 1L, 1L, 1L, 20L, 1L))
ssm0 <- sunspot.m2014[1:2988]
with(as.data.frame(.sunspot.diff), ssm0[i] <- ssm0[i] - res10/10)
sunspot.month.0 <- ts(ssm0, start = 1749, frequency = 12)
stopifnot(length(sunspot.month.0) == 2988)

```

sunspot.year

[sunspot.month](#)

sunspot.year

sunspot.year"ts"

[sunspot.monthsunspots](#)

<https://www.sidc.be/SILSO/datafiles>


```

utils::str(sm <- sunspots)# the monthly version we keep unchanged
utils::str(sy <- sunspot.year)
## The common time interval
(t1 <- c(max(start(sm), start(sy)), 1)) # Jan 1749
(t2 <- c(min( end(sm)[1],end(sy)[1]), 12)) # Dec 1983 (will not be updated!)
s.m <- window(sm, start=t1, end=t2)
s.y <- window(sy, start=t1, end=t2[1])
stopifnot(length(s.y) * 12 == length(s.m),
          ## The yearly series *is* close to the averages of the monthly one:
          all.equal(s.y, aggregate(s.m, FUN = mean), tolerance = 0.0020))
## NOTE: Strangely, correctly weighting the number of days per month
##       (using 28.25 for February) is *not* closer than the simple mean:
ndays <- c(31, 28.25, rep(c(31,30, 31,30, 31), 2))
all.equal(s.y, aggregate(s.m, FUN = mean)) # 0.0013
all.equal(s.y, aggregate(s.m, FUN = weighted.mean, w = ndays)) # 0.0017

```

sunspots

sunspots

[sunspot.monthsunspot.year](#)

```

require(graphics)
plot(sunspots, main = "sunspots data", xlab = "Year",
     ylab = "Monthly sunspot numbers")

```

swiss

swiss

[0, 100]

Fertility I_g
Agriculture
Examination
Education
Catholic
Infant.Mortality

Fertility

[0, 100]Catholic[0, 1]

<https://oprdata.princeton.edu/archive/pefp/switz.aspx>
ExaminationEducation

```
require(stats); require(graphics)
pairs(swiss, panel = panel.smooth, main = "swiss data",
      col = 3 + (swiss$Catholic > 50))
summary(lm(Fertility ~ . , data = swiss))
```

Theoph

Theoph

Theoph

```
c("nfnGroupedData", "nfGroupedData", "groupedData", "data.frame")
```

112

SSfol

[as.data.frame]plotprint

SSfol

```
require(stats); require(graphics)

coplot(conc ~ Time | Subject, data = Theoph, show.given = FALSE)
Theoph.4 <- subset(Theoph, Subject == 4)
fm1 <- nls(conc ~ SSfol(Dose, Time, lKe, lKa, lCl),
           data = Theoph.4)
summary(fm1)
plot(conc ~ Time, data = Theoph.4,
     xlab = "Time since drug administration (hr)",
     ylab = "Theophylline concentration (mg/L)",
     main = "Observed concentrations and fitted model",
     sub = "Theophylline data - Subject 4 only",
     las = 1, col = 4)
xvals <- seq(0, par("usr")[2], length.out = 55)
lines(xvals, predict(fm1, newdata = list(Time = xvals)),
      col = 4)
```

Titanic

Titanic

Class
Sex
Age
Survived

<https://www.encyclopedia-titanica.org/>

```
require(graphics)
mosaicplot(Titanic, main = "Survival on the Titanic")
## Higher survival rates in children?
apply(Titanic, c(3, 4), sum)
## Higher survival rates in females?
apply(Titanic, c(2, 4), sum)
## Use loglm() in package 'MASS' for further analysis ...
```

ToothGrowth

VC

ToothGrowth

len
supp
dose

```
require(graphics)
coplot(len ~ dose | supp, data = ToothGrowth, panel = panel.smooth,
       xlab = "ToothGrowth data: length vs dose, given type of supplement")
```

treering

treering

"ts"

<https://robjhyndman.com/TSDL/CA535.DAT>

<https://web.archive.org/web/20110523225828/http://www.ltrr.arizona.edu/~hallman/sitephotos/meth.html>

trees

trees

```
[,1] Girth
[,2] Height
[,3] Volume
```

```
require(stats); require(graphics)
pairs(trees, panel = panel.smooth, main = "trees data")
plot(log(Volume) ~ log(Girth), data = trees, log = "xy")
coplot(log(Volume) ~ log(Girth) | Height, data = trees,
       panel = panel.smooth)
summary(fm1 <- lm(log(Volume) ~ log(Girth), data = trees))
summary(fm2 <- update(fm1, ~ . + log(Height), data = trees))
step(fm2)
## i.e., Volume ~ c * Height * Girth^2 seems reasonable
```

UCBAdmissions

UCBAdmissions

```
Admit
Gender
Dept
```

k

```

require(graphics)
## Data aggregated over departments
apply(UCBAdmissions, c(1, 2), sum)
mosaicplot(apply(UCBAdmissions, c(1, 2), sum),
            main = "Student admissions at UC Berkeley")
## Data for individual departments
opar <- par(mfrow = c(2, 3), oma = c(0, 0, 2, 0))
for(i in 1:6)
  mosaicplot(UCBAdmissions[,i],
             xlab = "Admit", ylab = "Sex",
             main = paste("Department", LETTERS[i]))
mtext(expression(bold("Student admissions at UC Berkeley")),
       outer = TRUE, cex = 1.5)
par(opar)

```

UKDriverDeaths

UKDriverDeaths
Seatbelts

UKDriverDeaths
Seatbelts

Seatbelts

DriversKilled
drivers UKDriverDeaths
front
rear
kms
PetrolPrice
VanKilled
law

```

require(stats); require(graphics)
## work with pre-seatbelt period to identify a model, use logs
work <- window(log10(UKDriverDeaths), end = 1982+11/12)
par(mfrow = c(3, 1))
plot(work); acf(work); pacf(work)
par(mfrow = c(1, 1))
(fit <- arima(work, c(1, 0, 0), seasonal = list(order = c(1, 0, 0))))
z <- predict(fit, n.ahead = 24)
ts.plot(log10(UKDriverDeaths), z$pred, z$pred+2*z$se, z$pred-2*z$se,
        lty = c(1, 3, 2, 2), col = c("black", "red", "blue", "blue"))

## now see the effect of the explanatory variables
X <- Seatbelts[, c("kms", "PetrolPrice", "law")]
X[, 1] <- log10(X[, 1]) - 4
arima(log10(Seatbelts[, "drivers"]), c(1, 0, 0),
      seasonal = list(order = c(1, 0, 0)), xreg = X)

```

UKgas

UKgas

```
## maybe str(UKgas) ; plot(UKgas) ...
```

UKLungDeaths

ldeaths mdeaths fdeaths

ldeaths
fdeaths
mdeaths


```
require(stats); require(graphics) # for time
plot(ldeaths)
plot(mdeaths, fdeaths)
## Better labels:
yr <- floor(tt <- time(mdeaths))
plot(mdeaths, fdeaths,
      xy.labels = paste(month.abb[12*(tt - yr)], yr-1900, sep = "'"))
```

USAccDeaths

USAccDeaths

USArrests

USArrests

Murder
Assault
UrbanPop
Rape

USArrestsUrbanPopround()

<https://books.google.ch/books?id=z19qAAAAAAAJ&pg=PA20>

state

```
summary(USArrests)

require(graphics)
pairs(USArrests, panel = panel.smooth, main = "USArrests data")

## Difference between 'USArrests' and its correction
USArrests["Maryland", "UrbanPop"] # 67 -- the transcription error
UA.C <- USArrests
UA.C["Maryland", "UrbanPop"] <- 76.6

## also +/- 0.5 to restore the original <n>.5 percentages
s5u <- c("Colorado", "Florida", "Mississippi", "Wyoming")
s5d <- c("Nebraska", "Pennsylvania")
UA.C[s5u, "UrbanPop"] <- UA.C[s5u, "UrbanPop"] + 0.5
UA.C[s5d, "UrbanPop"] <- UA.C[s5d, "UrbanPop"] - 0.5

## ==> UA.C is now a *C*orrected version of USArrests
```

USJudgeRatings

USJudgeRatings

CONT
INTG
DMNR
DILG
CFMG
DECI
PREP
FAMI
ORAL
WRIT
PHYS
RTEN

```
require(graphics)
pairs(USJudgeRatings, main = "USJudgeRatings data")
```

USPersonalExpenditure

USPersonalExpenditure

```
require(stats) # for medpolish
USPersonalExpenditure
medpolish(log10(USPersonalExpenditure))
```

uspop

uspop

```
require(graphics)
plot(uspop, log = "y", main = "uspop data", xlab = "Year",
      ylab = "U.S. Population (millions)")
```

VADeaths

VADeaths

```
require(stats); require(graphics)
n <- length(dr <- c(VADeaths))
nam <- names(VADeaths)
d.VAD <- data.frame(
  Drate = dr,
  age = rep(ordered(rownames(VADeaths)), length.out = n),
  gender = gl(2, 5, n, labels = c("M", "F")),
  site = gl(2, 10, labels = c("rural", "urban")))
coplot(Drate ~ as.numeric(age) | gender * site, data = d.VAD,
  panel = panel.smooth, xlab = "VADeaths data - Given: gender")
summary(aov.VAD <- aov(Drate ~ .^2, data = d.VAD))
opar <- par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0))
plot(aov.VAD)
par(opar)
```

volcano

volcano

[filled.contour](#)

```
require(grDevices); require(graphics)
filled.contour(volcano, color.palette = terrain.colors, asp = 1)
title(main = "volcano data: filled contour map")
```

warpbreaks

warpbreaks

[,1]	breaks
[,2]	wool
[,3]	tension

ALAMAHBLMBH

[xtabs](#)

```
require(stats); require(graphics)
summary(warpbreaks)
opar <- par(mfrow = c(1, 2), oma = c(0, 0, 1.1, 0))
plot(breaks ~ tension, data = warpbreaks, col = "lightgray",
     varwidth = TRUE, subset = wool == "A", main = "Wool A")
plot(breaks ~ tension, data = warpbreaks, col = "lightgray",
     varwidth = TRUE, subset = wool == "B", main = "Wool B")
mtext("warpbreaks data", side = 3, outer = TRUE)
par(opar)
summary(fm1 <- lm(breaks ~ wool*tension, data = warpbreaks))
anova(fm1)
```

women

women

[,1] height
[,2] weight

```
require(graphics)
plot(women, xlab = "Height (in)", ylab = "Weight (lb)",
      main = "women data: American women aged 30-39")
```

WorldPhones

WorldPhones

```

require(graphics)
matplot(rownames(WorldPhones), WorldPhones, type = "b", log = "y",
        xlab = "Year", ylab = "Number of telephones (1000's)")
legend(1951.5, 80000, colnames(WorldPhones), col = 1:6, lty = 1:5,
       pch = rep(21, 7))
title(main = "World phones data: log scale for response")

```

WWWusage

WWWusage

```

require(graphics)
work <- diff(WWWusage)
par(mfrow = c(2, 1)); plot(WWWusage); plot(work)
## Not run:
require(stats)
aics <- matrix(, 6, 6, dimnames = list(p = 0:5, q = 0:5))
for(q in 1:5) aics[1, 1+q] <- arima(WWWusage, c(0, 1, q),
    optim.control = list(maxit = 500))$aic
for(p in 1:5)
    for(q in 0:5) aics[1+p, 1+q] <- arima(WWWusage, c(p, 1, q),
        optim.control = list(maxit = 500))$aic
round(aics - min(aics, na.rm = TRUE), 2)

## End(Not run)

```

grDevices

grDevices-package

`library(help = "grDevices")`

`<R-core@r-project.org>`

`adjustcolor`

(r, g, b, α)

```
adjustcolor(col, alpha.f = 1, red.f = 1, green.f = 1, blue.f = 1,
            offset = c(0, 0, 0, 0),
            transform = diag(c(red.f, green.f, blue.f, alpha.f)))
```

`col` `col2rgb()`

`alpha.f`

`red.f``green.f``blue.f`

`offset` `x := c(r,g,b,alpha)x[0,1]col2rgb(col, alpha=TRUE)`

`transform` `x + offset`

`colrgb()`

`rgbcol2rgbconvertColor`

```
## Illustrative examples :
opal <- palette("default")
stopifnot(identical(adjustcolor(1:8,      0.75),
                    adjustcolor(palette(), 0.75)))
cbind(palette(), adjustcolor(1:8, 0.75))

## alpha = 1/2 * previous alpha --> opaque colors
x <- palette(adjustcolor(palette(), 0.5))

sines <- outer(1:20, 1:4, function(x, y) sin(x / 20 * pi * y))
matplot(sines, type = "b", pch = 21:23, col = 2:5, bg = 2:5,
        main = "Using an 'opaque ('translucent') color palette")

x. <- adjustcolor(x, offset = c(0.5, 0.5, 0.5, 0), # <- "more white"
                 transform = diag(c(.7, .7, .7, 0.6)))
cbind(x, x.)
op <- par(bg = adjustcolor("goldenrod", offset = -rep(.4, 4)), xpd = NA)
plot(0:9, 0:9, type = "n", axes = FALSE, xlab = "", ylab = "",
     main = "adjustcolor() -> translucent")
text(1:8, labels = paste0(x,"++"), col = x., cex = 8)
par(op)

## and

(M <- cbind( rbind( matrix(1/3, 3, 3), 0), c(0, 0, 0, 1)))
adjustcolor(x, transform = M)

## revert to previous palette: active
palette(opal)
```

`as.graphicsAnnot`

`as.graphicsAnnot(x)`

`x`

`is.objectas.character`

as.raster

```
is.raster(x)
as.raster(x, ...)

## S3 method for class 'matrix'
as.raster(x, max = 1, ...)
## S3 method for class 'array'
as.raster(x, max = 1, ...)

## S3 method for class 'logical'
as.raster(x, max = 1, ...)
## S3 method for class 'numeric'
as.raster(x, max = 1, ...)
## S3 method for class 'character'
as.raster(x, max = 1, ...)
## S3 method for class 'raw'
as.raster(x, max = 255L, ...)
```

```
x
max
...
```

```
"raster"rgb
as.raster()
as.raster()
```

```
matrix
```

```
is.naNA
```

```
as.raster()
is.raster()x
```

```

# A red gradient
as.raster(matrix(hcl(0, 80, seq(50, 80, 10)),
                 nrow = 4, ncol = 5))

# Vectors are 1-column matrices ...
#   character vectors are color names ...
as.raster(hcl(0, 80, seq(50, 80, 10)))
#   numeric vectors are greyscale ...
as.raster(1:5, max = 5)
#   logical vectors are black and white ...
as.raster(1:10 %% 2 == 0)

# ... unless nrow/ncol are supplied ...
as.raster(1:10 %% 2 == 0, nrow = 1)

# Matrix can also be logical or numeric (or raw) ...
as.raster(matrix(c(TRUE, FALSE), nrow = 3, ncol = 2))
as.raster(matrix(1:3/4, nrow = 3, ncol = 4))

# An array can be 3-plane numeric (R, G, B planes) ...
as.raster(array(c(0:1, rep(0.5, 4)), c(2, 1, 3)))

# ... or 4-plane numeric (R, G, B, A planes)
as.raster(array(c(0:1, rep(0.5, 6)), c(2, 1, 4)))

# subsetting
r <- as.raster(matrix(colors()[1:100], ncol = 10))
r[, 2]
r[2:4, 2:5]

# assigning to subset
r[2:4, 2:5] <- "white"

# comparison
r == "white"

```

axisTicks

```

axisTicks(usr, log, axp = NULL, nint = 5)
.axisPars(usr, log = FALSE, nintLog = 5)

usr          c(min, max)par("usr")log = TRUElog10()
log
axp          c(mi, ma, n.)par("?axp")?xyn.
nintnintLog  nintLoglog = TRUE

```

```
axisTicks(usr, *).axisPars(usr, ..)axpNULL
axisTicks()CreateAtVector()/src/main/plot.caxis(side, *)at
CreateAtVector()
```

```
axisTicks()log=TRUEnint+1
.axisPars()list
axp          c(min., max.)
n            par("?axp")[3]
```

axTicksaxispar

```
##--- Demonstrating correspondence between graphics'
##--- axis() and the graphics-engine agnostic axisTicks() :

require("graphics")
plot(10*(0:10)); (pu <- par("usr"))
aX <- function(side, at, ...)
  axis(side, at = at, labels = FALSE, lwd.ticks = 2, col.ticks = 2,
        tck = 0.05, ...)
aX(1, print(xa <- axisTicks(pu[1:2], log = FALSE))) # x axis
aX(2, print(ya <- axisTicks(pu[3:4], log = FALSE))) # y axis

axisTicks(pu[3:4], log = FALSE, nint = 10)

## ----- Log Scale -----
x <- c(10, 1000)
# -----
axisTicks(log10(x), log = TRUE) # 10 20 50 .... 1000

plot(10*(0:10), log = "y"); (pu <- par("usr")) # ... .. 0.96 2.04
aX(2, print(ya <- axisTicks(pu[3:4], log = TRUE))) # 10 20 50 100 (y axis)

plot(2^(0:9), log = "y"); (pu <- par("usr"))
aX(2, print(ya <- axisTicks(pu[3:4], log = TRUE))) # y axis
## 'usr' corresponds to log10(<range>) :
stopifnot(ya == axisTicks(log10(c(1, 512)), log=TRUE))
```

boxplot.stats

```
boxplot.stats(x, coef = 1.5, do.conf = TRUE, do.out = TRUE)
```

```

x          NANA
coef       coefcoef
do.confdo.out FALSEconfout

```

```

quantile(x, c(1,3)/4)nn <- length(x)nn %% 4 == 1n ≡ 1 mod 4n %% 4 == 2n ≡ 2 mod 4
+/-1.58 IQR/sqrt(n)

```

```

stats      coef = 0fivenum(x, na.rm = TRUE)
n          NA
conf       if(do.conf)
out        if(do.out)

statsconfnout+- Inf

```

fivenumbboxplotbxbp

```

require(stats)
x <- c(1:100, 1000)
(b1 <- boxplot.stats(x))
(b2 <- boxplot.stats(x, do.conf = FALSE, do.out = FALSE))
stopifnot(b1 $ stats == b2 $ stats) # do.out = FALSE is still robust
boxplot.stats(x, coef = 3, do.conf = FALSE)

## no outlier treatment:
(b3 <- boxplot.stats(x, coef = 0))
stopifnot(b3$stats == fivenum(x))

## missing values are ignored
stopifnot(identical(boxplot.stats(c(x, NA)), b1))
## ... infinite values are not:
(r <- boxplot.stats(c(x, -1:1/0)))
stopifnot(r$out == c(1000, -Inf, Inf))

```

bringToTop

bringToTop-1
stay = TRUEstay

bringToTop(which = dev.cur(), stay = FALSE)

which -1
stay

[msgWindowwindows](#)

cairo

```
svg(filename = if(onefile) "Rplots.svg" else "Rplot%03d.svg",
     width = 7, height = 7, pointsize = 12,
     onefile = FALSE, family = "sans", bg = "white",
     antialias = c("default", "none", "gray", "subpixel"),
     symbolfamily)

cairo_pdf(filename = if(onefile) "Rplots.pdf" else "Rplot%03d.pdf",
          width = 7, height = 7, pointsize = 12,
          onefile = TRUE, family = "sans", bg = "white",
          antialias = c("default", "none", "gray", "subpixel"),
          fallback_resolution = 300, symbolfamily)

cairo_ps(filename = if(onefile) "Rplots.ps" else "Rplot%03d.ps",
          width = 7, height = 7, pointsize = 12,
          onefile = TRUE, family = "sans", bg = "white",
          antialias = c("default", "none", "gray", "subpixel"),
          fallback_resolution = 300, symbolfamily)
```

```
filename      PATH_MAXpdf
width
height
pointsize
onefile
family        "sans" "serif" "mono"
              X11
bg            par("bg")
antialias      "default"
fallback_resolution

symbolfamily
```

```
https://www.w3.org/Graphics/SVG/svgonefile = FALSE
pdfpostscriptcairo_pdfcairo_ps
cairo_ps(onefile = FALSE)

%dfilenameonefile = TRUE
cairo_ps
```

```
image
antialias = "default"antialias = "gray"
```

```
svg
```

```
capabilities("cairo")
```

```
Devicesdev.printpdfpostscript
capabilities
```

cairoSymbolFont

```
cairoSymbolFont(family, usePUA = TRUE)
```

```
family
```

```
usePUA
```

```
"CairoSymbolFont"
```

```
cairo_pdf
```

```
## Not run:  
## If a font uses PUA, we can just specify the font name ...  
cairo_pdf(symbolfamily="OpenSymbol")  
dev.off()  
## ... or equivalently ...  
cairo_pdf(symbolfamily=cairoSymbolFont("OpenSymbol"))  
dev.off()  
  
## If a font does not use PUA, we must indicate that ...  
cairo_pdf(symbolfamily=cairoSymbolFont("Nimbus Sans", usePUA=FALSE))  
dev.off()  
  
## End(Not run)
```

check.options

```
attributesnewname.opt
```

```
check.options(new, name.opt, reset = FALSE, assign.opt = FALSE,  
              envir = .GlobalEnv,  
              check.attributes = c("mode", "length"),  
              override.check = FALSE)
```



```
new
name.opt
reset      TRUEname.optname.optsearch()
assign.opt  TRUE
envir      environmentgetassign
check.attributes
            check.options
override.check length(new)new[i]override.check[i] == TRUE
```

```
name.optnewoverride.check
```

```
"names"NULL
```

```
ps.optionspdf.optionscheck.options
```

```
(L1 <- list(a = 1:3, b = pi, ch = "CH"))
check.options(list(a = 0:2), name.opt = "L1")
check.options(NULL, reset = TRUE, name.opt = "L1")
```

```
chull
```

```
chull(x, y = NULL)
```

```
xy          xyxxxy.coords
```

```
xy.coordsNaN
```

xy.coordspolygon

```
X <- matrix(stats::rnorm(2000), ncol = 2)
chull(X)

plot(X, cex = 0.5)
polygon(X[chull(X), ])

```

cm

cm(x)

x

cm(1) # = 2.54

Translate *from* cm *to* inches:

10 / cm(1) # -> 10cm are 3.937 inches

col2rgb

col2rgb(col, alpha = FALSE)

col colors()ipalette()[i]
alpha

```
NA"NA""transparent"
```

```
col
```

```
"#rrggbb""#rrggbbaa""#rgb""#rgba""#rgb""#rrggbb"
```

```
col
```

```
alpha = TRUEcolcol
```

```
rgbcolorspalette
```

```
convertColor()
```

```
col2rgb("peachpuff")
```

```
col2rgb(c(blu = "royalblue", reddish = "tomato")) # note: colnames
```

```
col2rgb(1:8) # the ones from the palette() (if the default)
```

```
col2rgb(paste0("gold", 1:4))
```

```
col2rgb("#08a0ff")
```

```
## all three kinds of color specifications:
```

```
col2rgb(c(red = "red", hex = "#abcdef"))
```

```
col2rgb(c(palette = 1:3))
```

```
# long and short form of hexadecimal notation
```

```
col2rgb(c(long = "#559955", short = "#595"))
```

```
# with alpha
```

```
col2rgb(c(long = "#559955BB", short = "#595B"), alpha = TRUE)
```

```
##-- NON-INTRODUCTORY examples --
```

```
grC <- col2rgb(paste0("gray", 0:100))
```

```
table(print(diff(grC["red",]))) # '2' or '3': almost equidistant
```

```
## The 'named' grays are in between {"slate gray" is not gray, strictly}
```

```
col2rgb(c(g66 = "gray66", darkg = "dark gray", g67 = "gray67",  
          g74 = "gray74", gray = "gray", g75 = "gray75",  
          g82 = "gray82", light = "light gray", g83 = "gray83"))
```

```
crgb <- col2rgb(cc <- colors())
```

```
colnames(crgb) <- cc
```

```
t(crgb) # The whole table
```

```
## How many names are 'aliases' of each other?
```

```
ccodes <- c(256^(2:0)) %%% crgb
```

```
cl <- split(cc, ccodes)
```

```
length(cl) # 502 distinct colors
```

```
table(tcc <- lengths(cl))
```

```
## All the multiply named colors:
```

```

clmult <- cl[tcc >= 2]
names(clmult) <- sapply(clmult, function(x) paste(crgb[,x[1]], collapse = ","))
utils::str(clmult)

## Look at the color cube:
tc <- t(crgb[, !duplicated(ccodes)])
cNms <- rownames(tc)
if(requireNamespace("lattice", quietly = TRUE))
  lattice::cloud(blue ~ red + green, data = as.data.frame(tc), col = cNms)
## The 8 corners of the color cube:
isC <- rowSums(tc == 0 | tc == 255) == 3
cNms[isC] # "white" "black" "blue" "cyan" "green" "magenta" "red" "yellow"

table(is.gray <- tc[,1] == tc[,2] & tc[,2] == tc[,3]) # (397, 105)

## Not run: ## Look at the color cube dynamically:
if(require("rgl")) {
  open3d(windowRect = c(50,50, 950, 950)) # large, so we see details
  plot3d(tc, col = cNms, size = 11) # --> rotate w/ mouse; enlarged corners:
  points3d(tc[isC,], col = cNms[isC], size=22)
  bg3d("darkgray") # (to "see more"); rotate around gray-axis:
  play3d(spin3d(axis = c(1, 1, 1), rpm = 2), duration = 30)
  if(FALSE) # add all names {zoom in with 2nd mouse button!}
    text3d(tc[!is.gray,], texts = cNms[!is.gray],
           col = cNms[!is.gray], adj=-1/4, cex = 1/2)
  if(FALSE) { ## next version of {rgl}
    hover3d(tc, labels = cNms)
    message("Move mouse over plot to identify points.")
  } else { ## click on blob to see colors()' name:
    identify3d(tc, labels=cNms)
  }
}

## End(Not run)

```

colorRamp

[topo.colors](#)[0,1]grey

```

colorRamp(colors, bias = 1, space = c("rgb", "Lab"),
          interpolate = c("linear", "spline"), alpha = FALSE)
colorRampPalette(colors, ...)

```

colors	col2rgb()
bias	
space	
interpolate	
alpha	space
...	colorRamp

```
space = "Lab"
```

```
colorRampfunction
```

```
colorRampPalettergbheat.colorsterrain.colors
```

```
splinefunapproxfun
```

```
## Both return a *function* :
colorRamp(c("red", "green"))( (0:4)/4 ) ## (x) , x in [0,1]
colorRampPalette(c("blue", "red"))( 4 ) ## (n)
## a ramp in opacity of blue values
colorRampPalette(c(rgb(0,0,1,1), rgb(0,0,1,0)), alpha = TRUE)(8)

require(graphics)

## Here space="rgb" gives palettes that vary only in saturation,
## as intended.
## With space="Lab" the steps are more uniform, but the hues
## are slightly purple.
filled.contour(volcano,
               color.palette =
                 colorRampPalette(c("red", "white", "blue")),
               asp = 1)
filled.contour(volcano,
               color.palette =
                 colorRampPalette(c("red", "white", "blue"),
                                   space = "Lab"),
               asp = 1)

## Interpolating a 'sequential' ColorBrewer palette
YlOrBr <- c("#FFFFD4", "#FED98E", "#FE9929", "#D95F0E", "#993404")
filled.contour(volcano,
               color.palette = colorRampPalette(YlOrBr, space = "Lab"),
               asp = 1)
filled.contour(volcano,
               color.palette = colorRampPalette(YlOrBr, space = "Lab",
                                                 bias = 0.5),
               asp = 1)

## 'jet.colors' is "as in Matlab"
## (and hurting the eyes by over-saturation)
jet.colors <-
  colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan",
                    "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
filled.contour(volcano, color.palette = jet.colors, asp = 1)

## space="Lab" helps when colors don't form a natural sequence
m <- outer(1:20,1:20,function(x,y) sin(sqrt(x*y)/3))
```

```

rgb.palette <- colorRampPalette(c("red", "orange", "blue"),
                               space = "rgb")
Lab.palette <- colorRampPalette(c("red", "orange", "blue"),
                               space = "Lab")
filled.contour(m, col = rgb.palette(20))
filled.contour(m, col = Lab.palette(20))

```

colors

```

colors (distinct = FALSE)
colours(distinct = FALSE)

```

```

distinct      "snow" "snow1" (0 : 255)3

```

```

col=
rgbhsvhclrainbowheat.colors
"transparent"

```

```

palettecol=
rgbhsvhclgrayrainbowheat.colorstopo.colors
col2rgb

```

```

cl <- colors()
length(cl); cl[1:20]

length(cl. <- colors(TRUE))
## only 502 of the 657 named ones

## ----- Show all named colors and more:
demo("colors")
## -----

```

contourLines

```
contourLines(x = seq(0, 1, length.out = nrow(z)),
             y = seq(0, 1, length.out = ncol(z)),
             z, nlevels = 10,
             levels = pretty(range(z, na.rm = TRUE), nlevels))
```

```
xy          zxlistx$xx$yxyz
z           NAxz
nlevels     levels
levels
```

```
contourLines
/src/main/plot3d.c
```

```
listlist
```

```
level
x
y
```

```
options("max.contour.segments")
contourcontourLines()contour()
```

```
x <- 10*1:nrow(volcano)
y <- 10*1:ncol(volcano)
cl <- contourLines(x, y, volcano)
## summarize the sizes of each the contour lines :
cbind(lev = vapply(cl, `[`, .5, "level"),
      n = vapply(cl, function(l) length(l$x), 1))
```

```
z <- outer(-9:25, -9:25)
pretty(range(z), 10) # -300 -200 ... 600 700
utils::str(c2 <- contourLines(z))
# no segments for {-300, 700};
# 2 segments for {-200, -100, 0}
# 1 segment for 100:600
```

convertColor

```
convertColor(color, from, to, from.ref.white, to.ref.white,  
             scale.in = 1, scale.out = 1, clip = TRUE)
```

```
color  
fromto  
from.ref.whiteto.ref.white  
          NULLD65  
scale.inscale.out  
          scale.inscale.outNULL  
clip          TRUEFALSENANa
```

```
colorConvertercolorConvertermake.rgb"XYZ""sRGB""Apple RGB""CIE RGB""Lab""Luv"  
colorspaces  
"sRGB""Apple RGB""Lab""Luv"XYZ  
LabLuvD65D50ABCEd55make.rgbD65"CIE RGB"  
clip  
colorfrom
```

<http://www.bruceindbloom.com/>

<https://web.archive.org/web/20190613001950/http://efg2.com/Lab/Graphics/Colors/Chromaticity.htm>

[col2rgbcolors](#)

[make.rgb](#)

```
## The displayable colors from four planes of Lab space  
ab <- expand.grid(a = (-10:15)*10,  
                 b = (-15:10)*10)  
require(graphics); require(stats) # for na.omit  
par(mfrow = c(2, 2), mar = .1+c(3, 3, 3, .5), mgp = c(2, .8, 0))  
  
Lab <- cbind(L = 20, ab)  
srgb <- convertColor(Lab, from = "Lab", to = "sRGB", clip = NA)
```



```

clipped <- attr(na.omit(srgb), "na.action")
srgb[clipped, ] <- 0
cols <- rgb(srgb[, 1], srgb[, 2], srgb[, 3])
image((-10:15)*10, (-15:10)*10, matrix(1:(26*26), ncol = 26), col = cols,
      xlab = "a", ylab = "b", main = "Lab: L=20")

Lab <- cbind(L = 40, ab)
srgb <- convertColor(Lab, from = "Lab", to = "sRGB", clip = NA)
clipped <- attr(na.omit(srgb), "na.action")
srgb[clipped, ] <- 0
cols <- rgb(srgb[, 1], srgb[, 2], srgb[, 3])
image((-10:15)*10, (-15:10)*10, matrix(1:(26*26), ncol = 26), col = cols,
      xlab = "a", ylab = "b", main = "Lab: L=40")

Lab <- cbind(L = 60, ab)
srgb <- convertColor(Lab, from = "Lab", to = "sRGB", clip = NA)
clipped <- attr(na.omit(srgb), "na.action")
srgb[clipped, ] <- 0
cols <- rgb(srgb[, 1], srgb[, 2], srgb[, 3])
image((-10:15)*10, (-15:10)*10, matrix(1:(26*26), ncol = 26), col = cols,
      xlab = "a", ylab = "b", main = "Lab: L=60")

Lab <- cbind(L = 80, ab)
srgb <- convertColor(Lab, from = "Lab", to = "sRGB", clip = NA)
clipped <- attr(na.omit(srgb), "na.action")
srgb[clipped, ] <- 0
cols <- rgb(srgb[, 1], srgb[, 2], srgb[, 3])
image((-10:15)*10, (-15:10)*10, matrix(1:(26*26), ncol = 26), col = cols,
      xlab = "a", ylab = "b", main = "Lab: L=80")

cols <- t(col2rgb(palette())); rownames(cols) <- palette(); cols
zapsmall(lab <- convertColor(cols, from = "sRGB", to = "Lab", scale.in = 255))
stopifnot(all.equal(cols, # converting back.. getting the original:
  round(convertColor(lab, from = "Lab", to = "sRGB", scale.out = 255)),
  check.attributes = FALSE))

```

densCols

densCols

```

densCols(x, y = NULL, nbin = 128, bandwidth,
         colramp = colorRampPalette(blues9[-(1:3)]))
blues9

```

xy	xyxy.coords
nbin	gridsizebkde2D()
bandwidth	bandwidthbkde2D
colramp	nn

```
densColsbkde2D(*, bandwidth, gridsize = nbin, ..)
blues9
```

```
densColsnrow(x)
```

```
bkde2DsmoothScatter()densCols
```

```
x1 <- matrix(rnorm(1e3), ncol = 2)
x2 <- matrix(rnorm(1e3, mean = 3, sd = 1.5), ncol = 2)
x  <- rbind(x1, x2)

dcols <- densCols(x)
graphics::plot(x, col = dcols, pch = 20, main = "n = 1000")
```

```
dev
```

```
dev.cur()
dev.list()
dev.next(which = dev.cur())
dev.prev(which = dev.cur())
dev.off(which = dev.cur())
dev.set(which = dev.next())
dev.new(..., noRStudioGD = FALSE)
graphics.off()
```

```
which
...
noRStudioGD    widthheight
```

```
"null device"getOption("device")
"X11""postscript""null device"dev.nextdev.prev
dev.offgraphics.off()graphics.off()
dev.setdev.nextwhich = 1
dev.newgetOption("device")pdfpostscriptRplots.pdfRplots1.pdfRplots999.pdfwidth
units = "in", res = 72widthheight
```

```
dev.cur
dev.listnamesNULL
dev.nextdev.prev
dev.off
dev.set
dev.newNULL
```

[Devicespostscript](#)
[layout](#)

```
## Not run: ## Unix-specific example
x11()
plot(1:10)
x11()
plot(rnorm(10))
dev.set(dev.prev())
abline(0, 1) # through the 1:10 points
dev.set(dev.next())
abline(h = 0, col = "gray") # for the residual plot
dev.set(dev.prev())
dev.off(); dev.off() #- close the two X devices

## End(Not run)
```

`dev.capabilities`

```
dev.capabilities(what = NULL)
```

```
what          NULL
```

```
NA
```

NA
 semiTransparency
 transparentBackground
 "no" "fully" "semi"
 rasterImage [rasterImagegrid.raster](#) "no" "yes" "non-missing"
 capture [grid.cap](#)
 locator [locatoridentify](#)
 events c("MouseDown", "MouseMove", "MouseUp", "Keybd", "Idle")
 patterns c("LinearGradient", "RadialGradient", "TilingPattern") FALSE
 clippingPaths
 masks c("alpha", "luminance") FALSE
 compositing [pdf](#) FALSE
 transformations

 paths
 glyphs

[getGraphicsEvent](#)

dev.capabilities()

dev.capture

dev.capture

dev.capture(native = FALSE)

native FALSE TRUE nativeRaster

NULL native = FALSE nativeRaster native = TRUE

`dev.flush`

```
dev.hold(level = 1L)
dev.flush(level = 1L)
```

```
level
```

```
dev.holddev.flushdev.holddev.flush
X11
quartz
windows
```

```
0
```

`dev.interactive`

```
dev.interactive(orNone = FALSE)
```

```
deviceIsInteractive(name = NULL)
```

```
orNone      TRUETRUE.Device == "null device"getOption("device")
name        NULL
```

```
X11windowsquartzJavaGDCairoWinCairoX11deviceIsInteractive
```

```
dev.interactive()TRUE
deviceIsInteractivename = NULL
```

[Devices](#)

```
dev.interactive()
print(deviceIsInteractive(NULL))
```

`dev.size`

```
dev.size(units = c("in", "cm", "px"))
```

```
units
```

```
dev.new()
```

```
par("din")par("din")dev.size
```

```
dev.size("cm")
```

`dev2`

```
dev.copywhichdevicewhichdevice
```

```
dev.printdevice
```

```
dev.copy2epsdev.printhehorizontal = FALSEdev.copy2pdf
```

```
dev.controldisplaylist"inhibit""enable"
```

```
dev.copy(device, ..., which = dev.next())
```

```
dev.print(device = postscript, ...)
```

```
dev.copy2eps(...)
```

```
dev.copy2pdf(..., out.type = "pdf")
```

```
dev.control(displaylist = c("inhibit", "enable"))
```

```
device          x11postscript
```

```
...             devicedev.copy2epspostscriptdev.copy2pdfpdfdev.printwhich  
                postscript
```

```
which
```

```
out.type        "pdf""quartz""cairo"cairo_pdf
```

```
displaylist     "inhibit""enable"
```

png

```
dev.copy2epsdev.copy2pdfwidthheightwidthheightRplot.epsRplot.pdf
pdfpostscriptfonts...familyfamily...
dev.printoptions("printcmd")postscriptdev.print(win.print)
dev.printfilewidthheightpointsizepointsizehorizontal
dev.printdevicepostscriptdev.copy2epspngjpegtifffbmunits = "inches"
```

```
dev.copy
dev.printdev.copy2epsdev.copy2pdf
```

```
dev.copy
dev.control("inhibit")dev.copydev.print
dev.control("inhibit")
```

dev.curdev.xxx

```
## Not run:
x11() # on a Unix-alike
plot(rnorm(10), main = "Plot 1")
dev.copy(device = x11)
mtext("Copy 1", 3)
dev.print(width = 6, height = 6, horizontal = FALSE) # prints it
dev.off(dev.prev())
dev.off()

## End(Not run)
```

dev2bitmap

bitmapdev2bitmap

```
bitmap(file, type = "png16m", height = 7, width = 7, res = 72,
        units = "in", pointsize, taa = NA, gaa = NA, ...)

dev2bitmap(file, type = "png16m", height = 7, width = 7, res = 72,
            units = "in", pointsize, ...,
            method = c("postscript", "pdf"), taa = NA, gaa = NA)
```

devAskNewPage

devAskNewPage(ask = NULL)

ask NULLTRUE

[options\("device.ask.default"\)](#)
[dev.control](#)

ask

[plot.newgrid.newpage](#)

Devices

[windows](#)
[pdf](#)
[postscript](#)
[xfig](#)
[bitmap](#) Ghostscript
[pictex](#)

[cairo_pdf](#)[cairo_ps](#)
[svg](#)
[png](#)
[jpeg](#)
[bmp](#)
[tiff](#)

[X11](#)
[quartz](#) R.app

```
options("device")pdfR_DEFAULT_DEVICEpdf()
```

```
https://pgf.sourceforge.net/
```

```
windows.options
```

```
X11.optionsquartz.options
```

```
ps.optionspdf.options
```

```
dev.interactivedev.curdev.printgraphics.offimagedev2bitmap
```

```
capabilitiesX11jpegpngtiffquartz
```

```
## Not run:
```

```
## open the default screen device on this platform if no device is
```

```
## open
```

```
if(dev.cur() == 1) dev.new()
```

```
## End(Not run)
```

```
embedFonts
```

```
embedGlyphs()glyphInfo
```

```
embedFonts(file, format, outfile = file,
```

```
          fontpaths = character(), options = character())
```

```
embedGlyphs(file, glyphInfo, outfile = file, options = character())
```

```
file
```

```
format          file
```

```
outfile
```

```
fontpaths
```

```
options
```

```
glyphInfo      glyphInfo()
```

```
options="-sFONTPATH=path/to/font"
ghostscriptR_GSCMD"gs"GSC"gswi64c.exe""gswin32c.exe"
format"ps2write".ps.eps"pdfwrite".pdfformat = "pswrite"format = "epswrite"format
= "eps2write"
```

```
-dMaxSubsetPct=100-dSubsetFonts=true-dEmbedAllFonts=true
embedGlyphs()pdf()glyphInfo
```

[postscriptFontsDevices](#)

https://www.r-project.org/doc/Rnews/Rnews_2006-2.pdf

extendrange

```
extendrange(x, r = range(x, na.rm = TRUE), f = 0.05)
```

```
x          r
r          rangex
f          f[1]f[2]
```

```
r + c(-f1,f2) * diff(r)f[1]f[2]f
```

[rangepretty](#)extendrange

```
x <- 1:5
(r <- range(x))      # 1    5
extendrange(x)       # 0.8  5.2
extendrange(x, f= 0.01) # 0.96 5.04

## extend more to the right:
extendrange(x, f=c(.01,.03)) # 0.96 5.12

## Use 'r' if you have it already:
stopifnot(identical(extendrange(r = r),
                    extendrange(x)))
```

getGraphicsEvent

```
getGraphicsEvent(prompt = "Waiting for input",
                 onMouseDown = NULL, onMouseMove = NULL,
                 onMouseUp = NULL, onKeybd = NULL,
                 onIdle = NULL,
                 consolePrompt = prompt)
setGraphicsEventHandlers(which = dev.cur(), ...)
getGraphicsEventEnv(which = dev.cur())
setGraphicsEventEnv(which = dev.cur(), env)
```

```
prompt
onMouseDown
onMouseMove
onMouseUp
onKeybd
onIdle
consolePrompt
which
...
env
```

```
windows()X11(type = "Xlib")X11(type = "cairo")
getGraphicsEventsetGraphicsEventHandlerssetGraphicsEventHandlers
setGraphicsEventEnvgetGraphicsEvent()getGraphicsEventEnv()
getGraphicsEventonMouseDownonMouseMoveonMouseUponKeybdonIdlepromptwhichresult
getGraphicsEvent()
function(buttons, x, y)xy(0,0)(1,1)buttons
function(key)shift-a"A"
  "Ctrl-A"
  "Left", "Up", "Right", "Down", "PgUp", "PgDn", "End", "Home"
  "Ins", "Del"
  "F1", "F2", ...
onIdleNULLonIdleX11()
Sys.sleep()Sys.sleep()Sys.sleep()getGraphicsEventsetGraphicsEventHandlers
```

```
NULLgetGraphicsEvent
```

```
getGraphicsEventNULLgetGraphicsEventNULLNULL
getGraphicsEventEnvNULL
setGraphicsEventEnvsetGraphicsEventHandlers
```

```
# This currently only works on the Windows, X11(type = "Xlib"), and
# X11(type = "cairo") screen devices...
## Not run:
savepar <- par(ask = FALSE)
dragplot <- function(..., xlim = NULL, ylim = NULL, xaxs = "r", yaxs = "r") {
  plot(..., xlim = xlim, ylim = ylim, xaxs = xaxs, yaxs = yaxs)
  startx <- NULL
  starty <- NULL
  prevx <- NULL
  prevy <- NULL
  usr <- NULL

  devset <- function()
    if (dev.cur() != eventEnv$which) dev.set(eventEnv$which)

  dragmousedown <- function(buttons, x, y) {
    startx <- x
    starty <- y
    prevx <- 0
    prevy <- 0
    devset()
    usr <- par("usr")
    eventEnv$onMouseMove <- dragmousemove
    NULL
  }

  dragmousemove <- function(buttons, x, y) {
    devset()
    deltax <- diff(grconvertX(c(startx, x), "ndc", "user"))
    deltay <- diff(grconvertY(c(starty, y), "ndc", "user"))
    if (abs(deltax-prevx) + abs(deltay-prevy) > 0) {
      plot(..., xlim = usr[1:2]-deltax, xaxs = "i",
        ylim = usr[3:4]-deltay, yaxs = "i")
      prevx <- deltax
      prevy <- deltay
    }
    NULL
  }

  mouseup <- function(buttons, x, y) {
    eventEnv$onMouseMove <- NULL
  }

  keydown <- function(key) {
    if (key == "q") return(invisible(1))
    eventEnv$onMouseMove <- NULL
  }
}
```

```

        NULL
    }

    setGraphicsEventHandlers(prompt = "Click and drag, hit q to quit",
                             onMouseDown = dragmousedown,
                             onMouseUp = mouseup,
                             onKeybd = keydown)
    eventEnv <- getGraphicsEventEnv()
}

dragplot(rnorm(1000), rnorm(1000))
getGraphicsEvent()
par(savepar)

## End(Not run)

```

glyphInfo

```

glyphInfo(id, x, y, font, size, fontList,
           width, height, hAnchor, vAnchor,
           col=NA, rot=0)

glyphFont(file, index, family, weight, style, PSname=NA)
glyphFontList(...)
glyphAnchor(value, label)
glyphWidth(w, label="width", left="left")
glyphHeight(h, label="height", bottom="bottom")
glyphWidthLeft(w, label)
glyphHeightBottom(h, label)
glyphJust(just, ...)
## S3 method for class 'GlyphJust'
glyphJust(just, ...)
## S3 method for class 'character'
glyphJust(just, ...)
## S3 method for class 'numeric'
glyphJust(just, which=NULL, ...)

```

```

id
xy
font          fontList
size
fontList      glyphFont()
width         glyphWidth()
height        glyphHeight()

```

```

hAnchor      xy"left"glyphAnchor()
vAnchor      xy"bottom"glyphAnchor()
col          NA
rot
file
index
family
weight
style        "normal""italic""oblique"
PSname       NA
valuewh
labelleftbottom

just         "left"0
which        x
...

```

```

xy"left""centre""right"hAnchor"left""left""bottom""centre""top"vAnchor"bottom"
"bottom""baseline"y
xy01"width""left""height""bottom""tight""left-bearing"
glyphWidthLeft()glyphWidthHeight()

```

```

glyphInfo()"RGlyphInfo"
glyphAnchor()glyphWidth()glyphHeight()widthheighthAnchorvAnchorglyphInfo()

```

```

NAidxsysizerot

```

```

gray

```

```

gray(level, alpha)
grey(level, alpha)

```

```

level        01"black""white"
alpha

```

graycol=[par](#)

greygray

level

[rainbowsvhclrgb](#)

gray(0:8 / 8)

gray.colors

n

gray.colors(n, start = 0.3, end = 0.9, gamma = 2.2, alpha, rev = FALSE)
grey.colors(n, start = 0.3, end = 0.9, gamma = 2.2, alpha, rev = FALSE)

n ≥ 1
start 01"black""white"
end
gamma
alpha
rev

gray.colorsnstartendseq(start^gamma, end^gamma, length = n)^(1/gamma)
[barplot.default](#)
grey.colorsgray.colors

n

[grayrainbowpalette](#)

require(graphics)

pie(rep(1, 12), col = gray.colors(12))
barplot(1:12, col = gray.colors(12))

grSoftVersion

grSoftVersion()

cairo ""

cairoFT "" "yes"

pango ""

libpng libpng""

jpeg ""

libtiff libtiff""

png(type = "cairo-png")

libjpeg-turbo"6.2"

extSoftVersion

grSoftVersion()

hcl

hcl(h = 0, c = 35, l = 85, alpha, fixup = TRUE)

h

c

l

alpha [0,1]

fixup fixupFALSENA

hclhsv

$0 < \alpha < 1$ rgb

hclNAalpha1

<https://www.R-project.org/conferences/DSC-2003/>

hsvrgb

```
require(graphics)

# The Foley and Van Dam PhD Data.
csd <- matrix(c( 4,2,4,6, 4,3,1,4, 4,7,7,1,
                 0,7,3,2, 4,5,3,2, 5,4,2,2,
                 3,1,3,0, 4,4,6,7, 1,10,8,7,
                 1,5,3,2, 1,5,2,1, 4,1,4,3,
                 0,3,0,6, 2,1,5,5), nrow = 4)

csphd <- function(colors)
  barplot(csd, col = colors, ylim = c(0,30),
          names.arg = 72:85, xlab = "Year", ylab = "Students",
          legend.text = c("Winter", "Spring", "Summer", "Fall"),
          main = "Computer Science PhD Graduates", las = 1)

# The Original (Metaphorical) Colors (Ouch!)
csphd(c("blue", "green", "yellow", "orange"))

# A Color Tetrad (Maximal Color Differences)
csphd(hcl(h = c(30, 120, 210, 300)))

# Same, but lighter and less colorful
# Turn off automatic correction to make sure
# that we have defined real colors.
csphd(hcl(h = c(30, 120, 210, 300),
            c = 20, l = 90, fixup = FALSE))
```

```

# Analogous Colors
# Good for those with red/green color confusion
csphd(hcl(h = seq(60, 240, by = 60)))

# Metaphorical Colors
csphd(hcl(h = seq(210, 60, length.out = 4)))

# Cool Colors
csphd(hcl(h = seq(120, 0, length.out = 4) + 150))

# Warm Colors
csphd(hcl(h = seq(120, 0, length.out = 4) - 30))

# Single Color
hist(stats::rnorm(1000), col = hcl(240))

## Exploring the hcl() color space {in its mapping to R's sRGB colors}:
demo(hclColors)

```

Hershey

```

familypar
textcontourvfont

```

Hershey

```

plotmath
par(family)par(font)

```

```

"HersheySerif"
"HersheySans"
"HersheyScript"
"HersheyGothicEnglish"
"HersheyGothicGerman"
"HersheyGothicItalian"
"HersheySymbol"
"HersheySansSymbol"

```

```

vfonttextcontourserifsans serifplainitalicvfontdemo(Hershey)a

```

typefacefontindexHershey(typeface, fontindex)

Hershey\$allowed

\\

\123\p"p""\160"

\\

\\ab*ademo(Hershey)

\\ab\366demo(Hershey)

U+00DFss

\\LI\\JUdemo(Hershey)

demo(Hershey)

("serif", fontindex)

\244\241

\\#J1234\\#J2421

\\#N1234\\#N0001

demo(Japanese)

\\#H1234\\#H0746demo(Hershey)

<https://www.gnu.org/software/plotutils/plotutils.html>

[demo\(Hershey\)](#)[partextcontour](#)
[Japanese](#)

Hershey

for tables of examples, see `demo(Hershey)`

hsv

`hsv(h = 1, s = 1, v = 1, alpha)`

hsv	[0, 1]
alpha	[0, 1]

$0 < \alpha < 1$ [rgb](#)

`hsvcol=par`

[hclhsv\(\)](#)[rgb2hsv](#)[rainbow](#)[gray](#)

`require(graphics)`

`hsv(.5,.5,.5)`

```
## Red tones:
n <- 20; y <- -sin(3*pi*((1:n)-1/2)/n)
op <- par(mar = rep(1.5, 4))
plot(y, axes = FALSE, frame.plot = TRUE,
      xlab = "", ylab = "", pch = 21, cex = 30,
      bg = rainbow(n, start = .85, end = .1),
      main = "Red tones")
par(op)
```

Japanese

Hershey

\\#J242b\\#J252b\\#J306c\\#N0001

demo(Japanese)

<https://www.gnu.org/software/plotutils/plotutils.html>

demo(Japanese)Hersheytext

```
require(graphics)

plot(1:9, type = "n", axes = FALSE, frame.plot = TRUE, ylab = "",
     main = "example(Japanese)", xlab = "using Hershey fonts")
par(cex = 3)
Vf <- c("serif", "plain")
text(4, 2, "\\#J244b\\#J245b\\#J2473", vfont = Vf)
text(4, 4, "\\#J2538\\#J2563\\#J2551\\#J2573", vfont = Vf)
text(4, 6, "\\#J467c\\#J4b5c", vfont = Vf)
text(4, 8, "Japan", vfont = Vf)
par(cex = 1)
text(8, 2, "Hiragana")
text(8, 4, "Katakana")
text(8, 6, "Kanji")
text(8, 8, "English")
```

make.rgb

convertColor

```
make.rgb(red, green, blue, name = NULL, white = "D65",
         gamma = 2.2)
```

```
colorConverter(toXYZ, fromXYZ, name, white = NULL, vectorized = FALSE)
```

```
redgreenblue
name
white
gamma          "sRGB"
fromXYZ
toXYZ
vectorized     fromXYZtoXYZ
```

```
gamma = "sRGB"
...
convertColorapplycolorConverterfromXYZtoXYZapplyfromXYZtoXYZvectorized=TRUE
```

```
colorConverter
```

```
http://www.brucelindbloom.com
```

```
convertColor
```

```
(pal <- make.rgb(red = c(0.6400, 0.3300),
  green = c(0.2900, 0.6000),
  blue = c(0.1500, 0.0600),
  name = "PAL/SECAM RGB"))

## converter for sRGB in #rrgbbb format
hexcolor <- colorConverter(toXYZ = function(hex, ...) {
  rgb <- t(col2rgb(hex))/255
  colorspace$sRGB$toXYZ(rgb, ...) },
  fromXYZ = function(xyz, ...) {
    rgb <- colorspace$sRGB$fromXYZ(xyz, ...)
    rgb <- round(rgb, 5)
    if (min(rgb) < 0 || max(rgb) > 1)
      as.character(NA)
    else rgb(rgb[1], rgb[2], rgb[3])},
  white = "D65", name = "#rrgbbb")

(cols <- t(col2rgb(palette()))))
zapsmall(luv <- convertColor(cols, from = "sRGB", to = "Luv", scale.in = 255))
(hex <- convertColor(luv, from = "Luv", to = hexcolor, scale.out = NULL))

## must make hex a matrix before using it
(cc <- round(convertColor(as.matrix(hex), from = hexcolor, to = "sRGB",
  scale.in = NULL, scale.out = 255)))
stopifnot(cc == cols)
```

```
## Internally vectorized version of hexcolor, notice the use
## of `vectorized = TRUE`:

hexcolorv <- colorConverter(toXYZ = function(hex, ...) {
  rgb <- t(col2rgb(hex))/255
  colorspace$sRGB$toXYZ(rgb, ...) },
  fromXYZ = function(xyz, ...) {
    rgb <- colorspace$sRGB$fromXYZ(xyz, ...)
    rgb <- round(rgb, 5)
    oob <- pmin(rgb[,1],rgb[,2],rgb[,3]) < 0 |
      pmax(rgb[,1],rgb[,2],rgb[,3]) > 0
    res <- rep(NA_character_, nrow(rgb))
    res[!oob] <- rgb(rgb[!oob,,drop=FALSE]),
    white = "D65", name = "#rrggbb",
    vectorized=TRUE)
(ccv <- round(convertColor(as.matrix(hex), from = hexcolor, to = "sRGB",
  scale.in = NULL, scale.out = 255)))
stopifnot(ccv == cols)
```

msgWindow

```
msgWindowwhich = -1
```

```
msgWindow(type = c("minimize", "restore", "maximize",
  "hide", "recordOn", "recordOff"),
  which = dev.cur())
```

```
type
which          -1
```

[bringToTopwindows](#)

n2mfrow	mfrow
---------	-------

```
par(mfrow)
```

```
n2mfrow(nr.plots, asp = 1)
```

```
nr.plots
asp          1aspnr.plots <= 12
```



```
(nr, nc)nr * nc >= nr.plotsasp = 1nr >= nc >= 1
```

```
nr >= 1nc >= 1nr.plots >= nr * ncnr >= asp*ncw(nr.plots - nr * nc) + w|nr/nc - asp|
```

```
asp
```

```
parlayout
```

```
require(graphics)
```

```
n2mfrow(8) # 3 x 3
```

```
n <- 5 ; x <- seq(-2, 2, length.out = 51)
```

```
## suppose now that 'n' is not known {inside function}
```

```
op <- par(mfrow = n2mfrow(n))
```

```
for (j in 1:n)
```

```
  plot(x, x^j, main = substitute(x^ exp, list(exp = j)), type = "l",  
       col = "blue")
```

```
sapply(1:14, n2mfrow)
```

```
sapply(1:14, n2mfrow, asp=16/9)
```

```
nclass
```

```
hist()
```

```
nclass.Sturges(x)
```

```
nclass.scott(x)
```

```
nclass.FD(x, digits = 5)
```

```
x
```

```
digits          xIQR
```

```
nclass.Sturges
```

```
nclass.scott1
```

```
nclass.FDIQR(signif(x, digits))digits = 5
```

L_2

[histtruehistdpih](#)

```
set.seed(1)
x <- stats::rnorm(1111)
nclass.Sturges(x)

## Compare them:
NC <- function(x) c(Sturges = nclass.Sturges(x),
  Scott = nclass.scott(x), FD = nclass.FD(x))
NC(x)
onePt <- rep(1, 11)
NC(onePt) # no longer gives NaN
```

palette

col=

```
palette(value)
palette.pals()
palette.colors(n = NULL, palette = "Okabe-Ito", alpha, recycle = FALSE,
  names = FALSE)
```

value

n [NULL](#)

palette [palette.pals\(\)](#)

alpha

recycle [n > length\(palette\(.\)\)recycle = FALSE](#)[n = NULL](#)

names

```
palette()palette.pals()palette.colors()
```

[par](#)

```
valuepalette.pals()"default"
```

value

value

[dev.copyreplayPlot](#)

```
palette()invisible
palette.pals()
palette.colors()names = FALSEnames = TRUE"Okabe-Ito""Tableau 10""Alphabet"
```

```
colorshsvgrayhcl.colors
adjustcolorcolorRampcol2rgb
```

```
require(graphics)

palette()          # obtain the current palette
palette("R3");palette() # old default palette
palette("ggplot2")   # ggplot2-style palette
palette()

palette(hcl.colors(8, "viridis"))

(palette(gray(seq(0,.9,length.out = 25)))) # gray scales; print old palette
matplot(outer(1:100, 1:30), type = "l", lty = 1,lwd = 2, col = 1:30,
        main = "Gray Scales Palette",
        sub = "palette(gray(seq(0, .9, len=25)))")
palette("default") # reset back to the default

## on a device where alpha transparency is supported,
## use 'alpha = 0.3' transparency with the default palette :
mycols <- adjustcolor(palette(), alpha.f = 0.3)
opal <- palette(mycols)
x <- rnorm(1000); xy <- cbind(x, 3*x + rnorm(1000))
plot (xy, lwd = 2,
      main = "Alpha-Transparency Palette\n alpha = 0.3")
xy[,1] <- -xy[,1]
points(xy, col = 8, pch = 16, cex = 1.5)
palette("default")

## List available built-in palettes
palette.pals()

## Demonstrate the colors 1:8 in different palettes using a custom matplot()
sinplot <- function(main=NULL, n = 8) {
  x <- outer(
    seq(-pi, pi, length.out = 50),
    seq( 0, pi, length.out = n),
    function(x, y) sin(x - y)
  )
  matplot(x, type = "l", lwd = 4, lty = 1, col = 1:n, ylab = "", main=main)
}
sinplot("default palette")

palette("R3");          sinplot("R3")
palette("Okabe-Ito"); sinplot("Okabe-Ito")
palette("Tableau") ; sinplot("Tableau", n = 10)
palROB <- colorRampPalette(c("red", "darkorange2", "blue"), space = "Lab")
```

```

palette(palROB(16)); sinplot("palROB(16)", n = 16)
palette("default") # reset

## color swatches for palette.colors()
palette.swatch <- function(palette = palette.pals(), n = 8, nrow = 8,
                           border = "black", cex = 1, ...)
{
  cols <- sapply(palette, palette.colors, n = n, recycle = TRUE)
  ncol <- ncol(cols)
  nswatch <- min(ncol, nrow)
  op <- par(mar = rep(0.1, 4),
            mfrow = c(1, min(5, ceiling(ncol/nrow))),
            cex = cex, ...)
  on.exit(par(op))
  while (length(palette)) {
    subset <- seq_len(min(nrow, ncol(cols)))
    plot.new()
    plot.window(c(0, n), c(0.25, nrow + 0.25))
    y <- rev(subset)
    text(0, y + 0.1, palette[subset], adj = c(0, 0))
    y <- rep(y, each = n)
    rect(rep(0:(n-1), n), y, rep(1:n, n), y - 0.5,
         col = cols[, subset], border = border)
    palette <- palette[-subset]
    cols <- cols[, -subset, drop = FALSE]
  }
}

palette.swatch()

palette.swatch(n = 26) # show full "Alphabet"; recycle most others

```

Palettes

`n`

```

hcl.colors(n, palette = "viridis", alpha = NULL, rev = FALSE, fixup = TRUE)
hcl.pals(type = NULL)

```

```

rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n,
        alpha, rev = FALSE)
heat.colors(n, alpha, rev = FALSE)
terrain.colors(n, alpha, rev = FALSE)
topo.colors(n, alpha, rev = FALSE)
cm.colors(n, alpha, rev = FALSE)

```

<code>n</code>	≥ 1
<code>palette</code>	<code>hcl.pals()</code>
<code>alpha</code>	<code>alpha</code> <code>missing</code> <code>alpha</code> <code>alpha = NULL</code> <code>"FF"</code>

```

rev
fixup      hcl
type       "qualitative""sequential""diverging""divergingx"NULL
sv
start
end

```

```

hcl.palsn
hsv
hcl

```

```

hcl.colors
hcl.pals

```

```

"Dark 3""YlGnBu""viridis""Green-Brown""Blue-Red 3"
palette.colors hcl.colors palette.colors
rainbowstartend  $\frac{1}{6} \frac{2}{6} \frac{3}{6} \frac{4}{6} \frac{5}{6}$  hcl.colors rainbow

```

```

cvhcl.palsn palette(cv) col = par

```

https://en.wikipedia.org/w/index.php?title=HCL_color_space&oldid=883465135

<https://www.R-project.org/conferences/DSC-2003/>

```

colorspalettegray.colors hsvhclrgbgraycol2rgb

```

```

require("graphics")

# color wheels in RGB/HSV and HCL space
par(mfrow = c(2, 2))
pie(rep(1, 12), col = rainbow(12), main = "RGB/HSV")
pie(rep(1, 12), col = hcl.colors(12, "Set 2"), main = "HCL")
par(mfrow = c(1, 1))

## color swatches for RGB/HSV palettes
demo.pal <-
  function(n, border = if (n < 32) "light gray" else NA,
           main = paste("color palettes; n=", n),

```

```

        ch.col = c("rainbow(n, start=.7, end=.1)", "heat.colors(n)",
                    "terrain.colors(n)", "topo.colors(n)",
                    "cm.colors(n)")
    {
        nt <- length(ch.col)
        i <- 1:n; j <- n / nt; d <- j/6; dy <- 2*d
        plot(i, i+d, type = "n", yaxt = "n", ylab = "", main = main)
        for (k in 1:nt) {
            rect(i-.5, (k-1)*j+ dy, i+.4, k*j,
                col = eval(str2lang(ch.col[k])), border = border)
            text(2*j, k * j + dy/4, ch.col[k])
        }
    }
demo.pal(16)

## color swatches for HCL palettes
hcl.swatch <- function(type = NULL, n = 5, nrow = 11,
    border = if (n < 15) "black" else NA) {
    palette <- hcl.pals(type)
    cols <- sapply(palette, hcl.colors, n = n)
    ncol <- ncol(cols)
    nswatch <- min(ncol, nrow)

    par(mar = rep(0.1, 4),
        mfrow = c(1, min(5, ceiling(ncol/nrow))),
        pin = c(1, 0.5 * nswatch),
        cex = 0.7)

    while (length(palette)) {
        subset <- 1:min(nrow, ncol(cols))
        plot.new()
        plot.window(c(0, n), c(0, nrow + 1))
        text(0, rev(subset) + 0.1, palette[subset], adj = c(0, 0))
        y <- rep(subset, each = n)
        rect(rep(0:(n-1), n), rev(y), rep(1:n, n), rev(y) - 0.5,
            col = cols[, subset], border = border)
        palette <- palette[-subset]
        cols <- cols[, -subset, drop = FALSE]
    }

    par(mfrow = c(1, 1), mar = c(5.1, 4.1, 4.1, 2.1), cex = 1)
}
hcl.swatch()
hcl.swatch("qualitative")
hcl.swatch("sequential")
hcl.swatch("diverging")
hcl.swatch("divergingx")

## heat maps with sequential HCL palette (purple)
image(volcano, col = hcl.colors(11, "purples", rev = TRUE))
filled.contour(volcano, nlevels = 10,
    color.palette = function(n, ...)
        hcl.colors(n, "purples", rev = TRUE, ...))

## list available HCL color palettes
hcl.pals("qualitative")
hcl.pals("sequential")

```

```
hcl.pals("diverging")
hcl.pals("divergingx")
```

pdf

pdf

```
pdf(file = if(onefile) "Rplots.pdf" else "Rplot%03d.pdf",
    width, height, onefile, family, title, fonts, version,
    paper, encoding, bg, fg, pointsize, pagecentre, colormodel,
    useDingbats, useKerning, fillOddEven, compress,
    timestamp, producer, author)
```

```
file
widthheight      7
onefile           TRUEfile
family            "Helvetica"
title             /Title"R Graphics Output"
fonts             NULL
version           "1.4"
paper             "a4"letter"legal"us"executive"a4r"USr"special"width
                  height"default"papersize"a4"special"
encoding          "default"
                  "ISOLatin1.enc"
                  "CP1250.enc"CP1251.enc"CP1253.enc"CP1257.enc"WinAnsi.enc"
                  enc".enc"
bg                "transparent"
fg                "black"
pointsize         12
pagecentre        paper != "special"TRUE
colormodel        "srgb"gray"grey"cmyk"srgb"
useDingbats       FALSETRUE

useKerning        TRUE
fillOddEven       polygonFALSE
compress          TRUE
timestamp         FALSE/CreationDate/ModDateTRUE
producer          FALSE/ProducerTRUE
author            /Author"
```

```

filepdf.options()
pdf()file
familyfonts
par(family = )pdfFonts
"Times""serif""Helvetica""sans""Courier""mono"embedFonts
xpdf

```

embedFonts

```

version
par(lwd = )pch = ". "cex = 1"cr"
paper/MediaBoxwidthheight"special"widthheight0.1postscriptpdf(paper = "a4r",
width = 0, height = 0)

```

path.expandfile

```

onefile = FALSE"Rplot%03d.pdf"Rplot001.pdfRplot999.pdfRplot1000.pdf
"%[#0 +--]*[0-9.]*[diouxX]"file%%%"
pdffileNULLstrwidthstringWidth

```

```

par(font = )gpar(fontface = )familypar(family = )gpar(fontfamily = )fonts
pdfFontspostscriptFonts
family"AvantGarde""Bookman""Courier""Helvetica""Helvetica-Narrow"
"NewCenturySchoolbook""Palatino""Times"
"URWGothic""URWBookman""NimbusMon""NimbusSan""NimbusSanCond""CenturySch"
"URWPalladio""NimbusRom""URWHelvetica" == "NimbusSan""URWTimes" == "NimbusRom"
embedFonts
"URW2Helvetica"Oblique"URW2HelveticaItalic"Italic"URW2Times""NimbusMonoPS"
"URWHelvetica""URWTimes""NimbusSan""NimbusRom""NimbusMon"

```

pdfFonts

```

familyType1FontCIDFont
embedFonts
useKerning = TRUEuseKerning = FALSE

```

textencoding

```

"TeXtext""ISOLatin2.enc""ISOLatin7.enc""ISOLatin9.enc""Cyrillic.enc"
"KOI8-R.enc""KOI8-U.enc""WinAnsi.enc""CP1252.enc""CP1250.enc""CP1251.enc"
"Greek.enc""CP1253.enc""CP1257.enc"
"ISOLatin1.enc"libiconv

"-""\u00ad"

```



```
"srgb""gray""grey""cmyk"https://en.wikipedia.org/wiki/CMYK\_color\_model#Mapping\_
RGB\_to\_CMYK
"rgb"
```

```
useDingbats = TRUE
```

```
pdf
```

```
pdf("|lp -o landscape", paper = "a4r")
```

```
onefile = TRUE
```

```
glyphInfoembedGlyphs
```

```
useDingbats = TRUEpch = 1"q"poppler~/.fonts.conf/etc/fonts/local.conf
```

```
<fontconfig>
<alias binding="same">
  <family>ZapfDingbats</family>
  <accept><family>Dingbats</family></accept>
</alias>
</fontconfig>
```

```
X11
```

```
pch = 1
```

```
pdf.jspch = 1
```

pdfFontspdf.optionseMBEDFontsglyphInfoDevicespostscript
cairo_pdfquartz

https://www.r-project.org/doc/Rnews/Rnews_2006-2.pdf

```
## Test function for encodings
TestChars <- function(encoding = "ISOLatin1", ...)
{
  pdf(encoding = encoding, ...)
  par(pty = "s")
  plot(c(-1,16), c(-1,16), type = "n", xlab = "", ylab = "",
       xaxs = "i", yaxs = "i")
  title(paste("Centred chars in encoding", encoding))
  grid(17, 17, lty = 1)
  for(i in c(32:255)) {
    x <- i %% 16
    y <- i %% 16
    points(x, y, pch = i)
  }
  dev.off()
}
## there will be many warnings.
TestChars("ISOLatin2")
## this does not view properly in older viewers.
TestChars("ISOLatin2", family = "URWHelvetica")
## works well for viewing in gs-based viewers, and often in xpdf.
```

pdf.options

pdf

pdf.optionspdf

pdf.optionspdfpdf

pdf.options(..., reset = FALSE)

... widthheighttonefilefamilytitlefontsversionpaperencodingbgfg
 pointsizepagecentrecolormodeluseDingbatsuseKerningfillOddEven
 compresstimestampproducerauthor

reset

reset = TRUE...

`pdfps.options`

```
pdf.options(bg = "pink")
utils::str(pdf.options())
pdf.options(reset = TRUE) # back to factory-fresh
```

`pictex`

```
pictex(file = "Rplots.tex", width = 5, height = 4, debug = FALSE,
        bg = "white", fg = "black")
```

```
file          path.expand
width
height
debug
bg
fg
```

`plotmath`
`pch = "."`

```
pointsize
cmss10
```

[pdfpostsriptDevices](#)

tikzDevice<https://pgf.sourceforge.net/pdftex>

```
require(graphics)
```

```
pictex()
plot(1:11, (-5:5)^2, type = "b", main = "Simple Example Plot")
dev.off()
##-----
## Not run:
## LaTeX Example
\documentclass{article}
\usepackage{pictex}
\usepackage{graphics} % for \rotatebox
\begin{document}
%...
\begin{figure}[h]
  \centerline{\input{Rplots.tex}}
  \caption{}
\end{figure}
%...
\end{document}

## End(Not run)
##-----
unlink("Rplots.tex")
```

plotmath

texttextmttextaxislegendpersp

demo(plotmath)

```

x + y
x - y
x*y
x/y
x %+-% y
x %/% y
x %*% y
x %.% y
x[i]
x^2
paste(x, y, z)
sqrt(x)
sqrt(x, y)
x == y
x != y
x < y
x <= y
x > y
x >= y
!x
x %~~% y
x %~% y
x %==% y
x %prop% y
x %~% y
plain(x)
bold(x)
italic(x)
bolditalic(x)
symbol(x)
list(x, y, z)
...
cdots
ldots
x %subset% y
x %subsepeq% y
x %notsubset% y
x %supset% y
x %supseteq% y
x %in% y
x %notin% y
hat(x)
tilde(x)
dot(x)
ring(x)
bar(xy)
widehat(xy)
widetilde(xy)
x %<->% y
x %->% y
x %<-% y

```

```

x %up% y
x %down% y
x %<=>% y
x %=>% y
x %<=% y
x %dblup% y
x %dbldown% y
alphaomega
AlphaOmega
theta1, phi1, sigma1, omega1
Upsilon1
aleph
infinity
partialdiff
nabla
32*degree
60*minute
30*second
displaystyle(x)
textstyle(x)
scriptstyle(x)
scriptscriptstyle(x)
underline(x)
x ~~ y
x + phantom(0) + y
x + over(1, phantom(0))
frac(x, y)
over(x, y)
atop(x, y)
sum(x[i], i==1, n)
prod(plain(P)(X==x), x)
integral(f(x)*dx, a, b)
union(A[i], i==1, n)
intersect(A[i], i==1, n)
lim(f(x), x %->% 0)
min(g(x), x > 0)
inf(S)
sup(S)
x^y + z
x^(y + z)
x^{y + z}
group("(",list(a, b),"]")
bgroup("(",atop(x,y),")")
group(lceil, x, rceil)
group(lfloor, x, rfloor)
group(langle, list(x, y), rangle)

```

| ([{ ". "" ||| lceil lfloor langle

paste

musymbol("m")symbol("\042")TestChars(font=5)points

`\Upsilon\Upsilonpsilon1\varepsilon\epsilon\epsilonpsilon\varpi\omega1\vartheta\theta\varphi\varsigma\sigma`
`\theta1\phi1\sigma1`

`\sigma1\sigma`

`\n`

`par(family=)gpar(fontfamily=)`

`bolditalicbolditalicmu`

`pointspdfpostscript`

`\uxxxx\UxxxxxxxxX11`

`quartz`

`\uxxxxpointswindows`

`RguiRterm`

<https://www.stat.auckland.ac.nz/~paul/R/CM/AdobeSym.html>

`demo(plotmath)axismtexttexttitlesubstitutequotebquote`

`require(graphics)`

`x <- seq(-4, 4, length.out = 101)`

`y <- cbind(sin(x), cos(x))`

`matplot(x, y, type = "l", xaxt = "n",
 main = expression(paste(plain(sin) * phi, " and ",
 plain(cos) * phi)),
 ylab = expression("sin" * phi, "cos" * phi), # only 1st is taken
 xlab = expression(paste("Phase Angle ", phi)),
 col.main = "blue")
axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
 labels = expression(-pi, -pi/2, 0, pi/2, pi))`

`## How to combine "math" and numeric variables :`

`plot(1:10, type="n", xlab="", ylab="", main = "plot math & numbers")`

`theta <- 1.23 ; mtext(bquote(hat(theta) == .(theta)), line= .25)`

`for(i in 2:9)`

`text(i, i+1, substitute(list(xi, eta) == group("(",list(x,y),")"),
 list(x = i, y = i+1)))`

`## note that both of these use calls rather than expressions.`

`##`

`text(1, 10, "Derivatives:", adj = 0)`

`text(1, 9.6, expression(
 " first: {f * minute}(x) " == {f * minute}(x)), adj = 0)`

`text(1, 9.0, expression(
 " second: {f * second}(x) " == {f * second}(x)), adj = 0)`

```

## note the "{ .. }" trick to get "chained" equations:
plot(1:10, 1:10, main = quote(1 <= {1 < 2}))
text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
text(4, 8.4, "expression(hat(beta) == (X^t * X)^{-1} * X^t * y)",
      cex = .8)
text(4, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))
text(4, 6.4, "expression(bar(x) == sum(frac(x[i], n), i==1, n))",
      cex = .8)
text(8, 5, expression(paste(frac(1, sigma*sqrt(2*pi)), " ",
                             plain(e)^{frac(-(x-mu)^2, 2*sigma^2)})),
      cex = 1.2)

## some other useful symbols
plot.new(); plot.window(c(0,4), c(15,1))
text(1, 1, "universal", adj = 0); text(2.5, 1, "\\042")
text(3, 1, expression(symbol("\\042")))
text(1, 2, "existential", adj = 0); text(2.5, 2, "\\044")
text(3, 2, expression(symbol("\\044")))
text(1, 3, "suchthat", adj = 0); text(2.5, 3, "\\047")
text(3, 3, expression(symbol("\\047")))
text(1, 4, "therefore", adj = 0); text(2.5, 4, "\\134")
text(3, 4, expression(symbol("\\134")))
text(1, 5, "perpendicular", adj = 0); text(2.5, 5, "\\136")
text(3, 5, expression(symbol("\\136")))
text(1, 6, "circlemultiply", adj = 0); text(2.5, 6, "\\304")
text(3, 6, expression(symbol("\\304")))
text(1, 7, "circleplus", adj = 0); text(2.5, 7, "\\305")
text(3, 7, expression(symbol("\\305")))
text(1, 8, "emptyset", adj = 0); text(2.5, 8, "\\306")
text(3, 8, expression(symbol("\\306")))
text(1, 9, "angle", adj = 0); text(2.5, 9, "\\320")
text(3, 9, expression(symbol("\\320")))
text(1, 10, "leftangle", adj = 0); text(2.5, 10, "\\341")
text(3, 10, expression(symbol("\\341")))
text(1, 11, "rightangle", adj = 0); text(2.5, 11, "\\361")
text(3, 11, expression(symbol("\\361")))

```

png

```

bmp(filename = "Rplot%03d.bmp",
     width = 480, height = 480, units = "px", pointsize = 12,
     bg = "white", res = NA, ...,
     type = c("cairo", "Xlib", "quartz"), antialias)

jpeg(filename = "Rplot%03d.jpeg",
     width = 480, height = 480, units = "px", pointsize = 12,
     quality = 75,
     bg = "white", res = NA, ...,

```



```

    type = c("cairo", "Xlib", "quartz"), antialias)

png(filename = "Rplot%03d.png",
     width = 480, height = 480, units = "px", pointsize = 12,
     bg = "white", res = NA, ...,
     type = c("cairo", "cairo-png", "Xlib", "quartz"), antialias)

tiff(filename = "Rplot%03d.tiff",
     width = 480, height = 480, units = "px", pointsize = 12,
     compression = c("none", "rle", "lzw", "jpeg", "zip",
                     "lzw+p", "zip+p",
                     "lerc", "lzma", "zstd", "webp"),
     bg = "white", res = NA, ...,
     type = c("cairo", "Xlib", "quartz"), antialias)

```

filename	PATH_MAXpdf
width	
height	
units	heightwidthpxincmm
pointsize	res
bg	par("bg")
quality	
compression	libtiff.tiff.htype = "quartz"
res	units
...	type = "Xlib"X11fontsfamily "cairo""quartz"familyX11 "cairo"symbolfamilyX11.options
type	"Xlib""quartz""cairo""Xlib"getOption("bitmapType")"quartz" "cairo""Xlib"
antialias	type = "cairo"X11X11.optionstype = "quartz"antialias = "none"

tiff

```

pngbg = "transparent" type = "Xlib" type = "cairo" type = "cairo-png" type = "quartz"
tiff"cairo""quartz""zip""lzw+p""zip+p"
jpeg.tiff
type = "Xlib" type = "cairo" type = "quartz"quartzcapabilities("aqua")

```

res

```
pngtype = "cairo" type = "cairo-png"
```

Preview

```
widthheight
%dfilename
```

```
fontconfig
```

```
.jpg.tif.jpeg.tiff
```

```
res
```

```
restype = "Xlib" type = "cairo"
type = "Xlib"
```

```
type = "quartz" quartz
```

```
type = "Xlib" X11 options
libtiff"lerc""lzma""zstd""webp"
```

```
https://www.w3.org/TR/png/
```

```
https://www.iso.org/standard/34342.htmlhttps://en.wikipedia.org/wiki/TIFF
```

```
Devicesdev.print
```

```
capabilities type = "cairo"
```

```
bitmap
```

```
## these examples will work only if the devices are available
## and cairo or an X11 display or a macOS display is available.

## copy current plot to a (large) PNG file
## Not run: dev.print(png, filename = "myplot.png", width = 1024, height = 768)

png("myplot.png", bg = "transparent")
plot(1:10)
rect(1, 5, 3, 7, col = "white")
dev.off()

## will make myplot1.jpeg and myplot2.jpeg
jpeg("myplot%d.jpeg")
example(rect)
dev.off()
```

postscript

postscript

```
postscript(file = if(onefile) "Rplots.ps" else "Rplot%03d.ps",
           onefile, family, title, fonts, encoding, bg, fg,
           width, height, horizontal, pointsize,
           paper, pagecentre, print.it, command,
           colormodel, useKerning, fillOddEven)
```

file	<pre>""command" cmd"cmd onefile = FALSE"Rplot%03d.ps"%%%"%["#0 +=-]*[0-9.]*[diouxX]" path.expand pdf</pre>
onefile	DocumentMediaTRUE
family	pdf"Helvetica"
title	Title"R Graphics Output"
fonts	NULL
encoding	pdf"default"
bg	"transparent""transparent"
fg	"black"
widthheight	<pre>0 paper != "special"widthheight0.1</pre>
horizontal	
pointsize	12
paper	<pre>"a4""letter""us""legal""executive""special"widthheight"default" "papersize""a4"</pre>

```

pagecentre
print.it      file
command      "default""printcmd"2*PATH_MAX
colormodel    "srgb""srgb+gray""rgb""rgb-nogray""gray""grey")"cmyk""srgb"
useKerning    TRUE
fillOddEven   polygonFALSE

```

```

fileps.options()
postscriptfile
filesprintfRplot001.psRplot999.psRplot1000.ps
\includegraphics{<filename>}setEPS()horizontal = FALSE, onefile = FALSE, paper
= "special"par(mar = )
.ps.prolognamespace:grDevices
familyfontspostscriptFonts
par(lwd = )pch = "."cex = 1"cra"

```

```

encoding = "TeXtext.enc"< > \ _ { }
"ComputerModern""ComputerModernItalic"postscriptpdfdvips -Ppfb -j0-j0family =
"ComputerModern"cmsl10cmbxsl10family = "ComputerModernItalic"
CM_symbol_10.afm

```

```

"srgb"
"srgb+gray"
"gray""grey""cmyk"https://en.wikipedia.org/wiki/CMYK\_color\_model#Mapping\_RGB\_
to\_CMYK
"rgb""rgb-nogray"

```

```

postscript

print.it = TRUEcommand"file"command
file = ""file = "|cmd"dev.off

"printcmd"print.it      =      TRUERedMonhttp://pages.cs.wisc.edu/~ghost/index.html
gsprint.exe

```

<durso@hussle.harvard.edu>

postscriptFontsDevicescheck.optionsps.optionspostscript
cairo_ps

https://www.r-project.org/doc/Rnews/Rnews_2006-2.pdf

```
require(graphics)
## Not run:
# open the file "foo.ps" for graphics output
postscript("foo.ps")
# produce the desired graph(s)
dev.off()          # turn off the postscript device

## On Unix-alikes only:
postscript("|lp -dlw")
# produce the desired graph(s)
dev.off()          # plot will appear on printer

## On Windows:
options(printcmd = 'redpr -P"\\printhost\\lw"')
postscript(file = tempfile("Rps."), print.it = TRUE)
# produce the desired graph(s)
dev.off()          # send plot file to the printer
## alternative using GSView 4.x :
options(printcmd = '/GhostGum/gsview/gsprint -query')
```

```

# for URW PostScript devices
postscript("foo.ps", family = "NimbusSan")

## for inclusion in Computer Modern TeX documents, perhaps
postscript("cm_test.eps", width = 4.0, height = 3.0,
           horizontal = FALSE, onefile = FALSE, paper = "special",
           family = "ComputerModern", encoding = "TeXtext.enc")
## The resultant postscript file can be used by dvips -Ppfb -j0.

## To test out encodings, you can use
TestChars <- function(encoding = "ISOLatin1", family = "URWHelvetica")
{
  postscript(encoding = encoding, family = family)
  par(pty = "s")
  plot(c(-1,16), c(-1,16), type = "n", xlab = "", ylab = "",
       xaxs = "i", yaxs = "i")
  title(paste("Centred chars in encoding", encoding))
  grid(17, 17, lty = 1)
  for(i in c(32:255)) {
    x <- i %% 16
    y <- i %/% 16
    points(x, y, pch = i)
  }
  dev.off()
}
## there will be many warnings. We use URW to get a complete enough
## set of font metrics.
TestChars()
TestChars("ISOLatin2")
TestChars("WinAnsi")

## End(Not run)

```

postscriptFonts

[postscriptpdf](#)

```

postscriptFonts(...)
pdfFonts(...)

```

...

```

postscript"family"par"fontfamily"gpar
postscriptpdf
postscriptFontspdfFontsType1FontCIDFontxxxFonts

```

"sans" "Helvetica" "serif" "Times" "mono" "Courier"
"AvantGarde" "Bookman" "Courier" "Helvetica" "Helvetica-Narrow"
"NewCenturySchoolbook" "Palatino" "Times" "URWGothic" "URWBookman" "NimbusMon"
"NimbusSan" "URWHelvetica" "NimbusSanCond" "CenturySch" "URWPalladio" "NimbusRom"
"URWTimes"
"ComputerModern" "ComputerModernItalic" "ArialMT"

postscript

.pfb.pfaembedFonts

postscriptFontsfamily

familyfonts

"Japan1" "Japan1HeiMin" "Japan1GothicBBB" "Japan1Ryumin" "Korea1" "Korea1deb" "GB1"
"CNS1"

HeiseiKakuGo-W5

KozMinPro-Regular-Acro

HeiseiMin-W3

HeiseiMin-W3-Acro

GothicBBB-Medium

Ryumin-Light

Baekmuk-Batang

HYSMyeongJoStd-Medium-Acro

Batang-Regular

HYGothic-Medium-Acro

BousungEG-Light-GB

STSong-Light-Acro

MOESung-Regular

MSungStd-Light-Acro

BousungEG-Light-GB<https://ftp.gnu.org/pub/non-gnu/chinese-fonts-truetype/>
-Acro

postscriptpdfType1FontCIDFont

```
postscriptFonts()
## This duplicates "ComputerModernItalic".
CMitalic <- Type1Font("ComputerModern2",
  c("CM_regular_10.afm", "CM_boldx_10.afm",
    "cmti10.afm", "cmbxti10.afm",
    "CM_symbol_10.afm"),
  encoding = "TeXtext.enc")
postscriptFonts(CMitalic = CMitalic)

## A CID font for Japanese using a different CMap and
## corresponding cmapEncoding.
`Jp_UCS-2` <- CIDFont("TestUCS2",
  c("Adobe-Japan1-UniJIS-UCS2-H.afm",
    "Adobe-Japan1-UniJIS-UCS2-H.afm",
    "Adobe-Japan1-UniJIS-UCS2-H.afm",
    "Adobe-Japan1-UniJIS-UCS2-H.afm"),
  "UniJIS-UCS2-H", "UCS-2")
pdfFonts(`Jp_UCS-2` = `Jp_UCS-2`)
names(pdfFonts())
```

pretty.Date

```
n+1xmin.n = 0x
```

```
## S3 method for class 'Date'
pretty(x, n = 5, min.n = n %/% 2, sep = " ", ...)
## S3 method for class 'POSIXt'
pretty(x, n = 5, min.n = n %/% 2, sep = " ", ...)
```

```
x          "Date""POSIXt""POSIXct""POSIXlt"
n
min.n
sep
...
```


"labels""format"

[pretty](#)

```
pretty(Sys.Date())
pretty(Sys.time(), n = 10)

pretty(as.Date("2000-03-01")) # R 1.0.0 came in a leap year

## time ranges in diverse scales:
steps <- stats::setNames(,
  c("10 secs", "1 min", "5 mins", "30 mins", "6 hours", "12 hours",
    "1 DSTday", "2 weeks", "1 month", "6 months", "1 year",
    "10 years", "50 years", "1000 years"))
x <- as.POSIXct("2002-02-02 02:02")
lapply(steps,
  function(s) {
    at <- pretty(seq(x, by = s, length.out = 2), n = 5)
    attr(at, "labels")
  })
```

ps.options

ps.options[postscript](#)

ps.optionspostscriptpostscript

ps.options(..., reset = FALSE, override.check = FALSE)

setEPS(...)

setPS(...)

... onefilefamilytitlefontsencodingbgfgwidthheighthorizontal
 pointsizepaperpagecentreprint.itcommandcolormodelfillOddEven
 onefilehorizontalpapersetEPSsetPS

reset

override.check [check.options](#)

reset = TRUE...

append

setEPSsetPSwidthheight

```
...reset = TRUE
```

postscriptpdf.options

```
ps.options(bg = "pink")
utils::str(ps.options())

### ---- error checking of arguments: ----
ps.options(width = 0:12, onefile = 0, bg = pi)
# override the check for 'width', but not 'bg':
ps.options(width = 0:12, bg = pi, override.check = c(TRUE,FALSE))
utils::str(ps.options())
ps.options(reset = TRUE) # back to factory-fresh
```

quartz

quartz

```
quartz(title, width, height, pointsize, family, antialias, type,
       file = NULL, bg, canvas, dpi)
```

```
quartz.options(..., reset = FALSE)
```

```
quartz.save(file, type = "png", device = dev.cur(), dpi = 100, ...)
```

title	"Quartz %d"filepdf
width	7
height	7
pointsize	12
family	"Arial"
antialias	TRUE
type	"native"
file	NULL
bg	"transparent""white""png""tiff"
canvas	"white"
dpi	NA_real_
...	quartzfile
reset	
device	

```
quartz::quartz.options(
  """"native""Cocoa""filetype    =    ""pdf""""png""""jpeg""""jpg""""jpeg2000""""tif""""tiff""""gif""
  ""psd""""bmp""""sgi""""pict""
  filedev.off()Rplot%03d.pngPATH_MAXpdfRplots.pdfRplot%03d.path.expand
  par(family =)quartzFontsHelveticaTimes-RomanserifCouriermono
  canvastype = ""png""type = ""tiff""Preview
  title
  quartz().Device""quartz""""quartz_off_screen""
  ""Arial""""MingLiU""
  quartz.savedev.copy2pdfquartz
```

quartzFontsDevices

png

quartzFonts

quartz

quartzFont(family)

quartzFonts(...)

family

...

quartzgpar

quartzFontsquartzFont

"sans""serif""mono"

quartz

```
if(.Platform$OS.type == "unix") { # includes macOS
```

```
  utils::str( quartzFonts() ) # a list(serif = .., sans = .., mono = ..)
  quartzFonts("mono")       # the list(mono = ..) sublist of quartzFonts()
```

```
## Not run:
```

```
## for East Asian locales you can use something like
quartzFonts(sans = quartzFont(rep("AppleGothic", 4)),
             serif = quartzFont(rep("AppleMyungjp", 4)))
## since the default fonts may well not have the glyphs needed
```

```
## End(Not run)
}
```

`recordGraphics`

```
recordGraphics(expr, list, env)
```

```
expr          expressioncall
list          expr
env           environmentlist
```

```
exprlistenv
```

```
expr
```

```
assign
```

```
eval
```

```
require(graphics)
```

```
plot(1:10)
# This rectangle remains 1inch wide when the device is resized
recordGraphics(
  {
    rect(4, 2,
          4 + diff(par("usr")[1:2])/par("pin")[1], 3)
  },
  list(),
  getNamespace("graphics"))
```

recordPlot

```
recordPlot(load=NULL, attach=NULL)
replayPlot(x, reloadPkgs=FALSE)
```

```
load          NULL
```

```
attach        NULL
```

```
x
```

```
reloadPkgs
```

```
"recordedplot"replayPlotprint
```

```
deparsestr
```

```
recordPlot"recordedplot"
```

```
replayPlot
```

```
saveRDSreadRDS
```

```
recordGraphicsloadattachrecordPlotloadloadNamespaceattachlibraryreloadPkgsTRUE
replayPlot
```

```
https://stattech.wordpress.fos.auckland.ac.nz/2015/12/21/2015-07-recording-and-replaying-the-graphics-engine-display-list/
```

```
dev.control
```

rgb

max

0maxalpha

names

col=par

rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)

redbluegreenalpha

$[0, M]$ M maxColorValue 255 redbluegreenalpha 0:255

names

maxColorValue

redbluegreenredredgreenblue

0 < alpha < 1 pdf windows quartz X11 (type = "cairo") jpeg png bmp tiff bitmap

NA redbluegreenalpha

"#" 0 ... 255 0 255

col2rgb rainbow hsv hcl gray

rgb(0, 1, 0)

rgb((0:15)/15, green = 0, blue = 0, names = paste("red", 0:15, sep = "."))

rgb(0, 0:12, 0, maxColorValue = 255) # integer input

ramp <- colorRamp(c("red", "white"))

rgb(ramp(seq(0, 1, length.out = 5)), maxColorValue = 255)

rgb2hsv

rgb2hsv

```
rgb2hsv(r, g = NULL, b = NULL, maxColorValue = 255)
```

```
r          [0, M] M = maxColorValue
g          NULL r
b          NULL r
maxColorValue 255 0:255 col2rgb()
```

"h"s"v"

<wolfram@fischer-zim.ch>
<nikko@hailmail.net>

hsvcol2rgb

```
## These (saturated, bright ones) only differ by hue
(rc <- col2rgb(c("red", "yellow", "green", "cyan", "blue", "magenta")))
(hc <- rgb2hsv(rc))
6 * hc["h",] # the hues are equispaced

(rgb3 <- floor(256 * matrix(stats::runif(3*12), 3, 12)))
(hsv3 <- rgb2hsv(rgb3))
## Consistency :
stopifnot(rgb3 == col2rgb(hsv(h = hsv3[1,], s = hsv3[2,], v = hsv3[3,])),
  all.equal(hsv3, rgb2hsv(rgb3/255, maxColorValue = 1)))

## A (simplified) pure R version -- originally by Wolfram Fischer --
## showing the exact algorithm:
rgb2hsvR <- function(rgb, gamma = 1, maxColorValue = 255)
{
  if(!is.numeric(rgb)) stop("rgb matrix must be numeric")
  d <- dim(rgb)
```



```

if(d[1] != 3) stop("rgb matrix must have 3 rows")
n <- d[2]
if(n == 0) return(cbind(c(h = 1, s = 1, v = 1))[,0])
rgb <- rgb/maxColorValue
if(gamma != 1) rgb <- rgb ^ (1/gamma)

## get the max and min
v <- apply( rgb, 2, max)
s <- apply( rgb, 2, min)
D <- v - s # range

## set hue to zero for undefined values (gray has no hue)
h <- numeric(n)
notgray <- ( s != v )

## blue hue
idx <- (v == rgb[3,] & notgray )
if (any (idx))
  h[idx] <- 2/3 + 1/6 * (rgb[1,idx] - rgb[2,idx]) / D[idx]
## green hue
idx <- (v == rgb[2,] & notgray )
if (any (idx))
  h[idx] <- 1/3 + 1/6 * (rgb[3,idx] - rgb[1,idx]) / D[idx]
## red hue
idx <- (v == rgb[1,] & notgray )
if (any (idx))
  h[idx] <- 1/6 * (rgb[2,idx] - rgb[3,idx]) / D[idx]

## correct for negative red
idx <- (h < 0)
h[idx] <- 1+h[idx]

## set the saturation
s[! notgray] <- 0;
s[notgray] <- 1 - s[notgray] / v[notgray]

rbind( h = h, s = s, v = v )
}

## confirm the equivalence:
all.equal(rgb2hsv (rgb3),
          rgb2hsvR(rgb3), tolerance = 1e-14) # TRUE

```

savePlot

X11()

```

savePlot(filename = paste0("Rplot.", type),
          type = c("png", "jpeg", "tiff", "bmp"),
          device = dev.cur())

```

filename
type
device

X11

`dev.hold`
`bg = "transparent"`

NULL

windowsrestoreConsole

`recordPlot()``X11dev.copydev.print`

trans3d

`persp`

`trans3d(x, y, z, pmat, continuous = FALSE, verbose = TRUE)`

xyz	
pmat	$4 \times 4(x, y, z)(x, y, z, t)$ <code>persp()</code>
continuous	$a/0(x, y, z)$ <code>continuous=TRUE</code>
verbose	<code>continuous=TRUE</code>

xy (x,y,z)

`persp`

```
## See help(persp) {after attaching the 'graphics' package}
## -----

## Example for 'continuous = TRUE' (vs default):
require(graphics)
x <- -10:10/10 # [-1, 1]
y <- -16:16/16 # [-1, 1] ==> z = fxy := outer(x,y) is also in [-1,1]

p <- persp(x, y, fxy <- outer(x,y), phi = 20, theta = 15, r = 3, ltheta = -75,
           shade = 0.8, col = "green3", ticktype = "detailed")
## 5 axis-parallel auxiliary lines in x-y and y-z planes :
lines(trans3d(-.5 , y=-1:1, z=min(fxy), pmat=p), lty=2)
lines(trans3d( 0 , y=-1:1, z=min(fxy), pmat=p), lty=2)
lines(trans3d(-1:1, y= -.7, z=min(fxy), pmat=p), lty=2)
lines(trans3d(-1, y= -.7, z=c(-1,1) , pmat=p), lty=2)
lines(trans3d(-1, y=-1:1, z= -.5 , pmat=p), lty=2)
## 2 pillars to carry the horizontals below:
lines(trans3d(-.5 , y= -.7, z=c(-1,-.5), pmat=p), lwd=1.5, col="gray10")
lines(trans3d( 0 , y= -.7, z=c(-1,-.5), pmat=p), lwd=1.5, col="gray10")
## now some "horizontal rays" (going from center to very left or very right):
doHor <- function(x1, x2, z, CNT=FALSE, ...)
  lines(trans3d(x=seq(x1, x2, by=0.5), y= -0.7, z = z, pmat = p, continuous = CNT),
        lwd = 3, type="b", xpd=NA, ...)
doHor(-10, 0, z = -0.5, col = 2) # x in [-10, 0] -- to the very left : fine
doHor(-.5, 2, z = -0.52,col = 4) # x in [-0.5, 2] only {to the right} --> all fine
## but now, x in [-0.5, 20] -- "too far" ==> "wrap around" problem (without 'continuous=TRUE'):
doHor(-.5, 20, z = -0.58, col = "steelblue", lty=2)
## but it is fixed with continuous = CNT = TRUE:
doHor(-.5, 20, z = -0.55, CNT=TRUE, col = "skyblue")
```

Type1Font

pdfpostscript

```
Type1Font(family, metrics, encoding = "default")
```

```
CIDFont(family, cmap, cmapEncoding, pdfresource = "")
```

```
family          postscriptFonts
```

```
metrics
```

```
cmap
```

```
encoding        Type1Font"default""ISOLatin1.enc""WinAnsi.enc"enc".enc"
```

```
cmapEncoding
```

```
pdfresource     pdf
```

```
Type1Fonts.afm"Symbol.afm"/library/grDevices/afmAdobeSym.afm
.afm.gz
cmapcmapEncodingpdfpostscript
plotmath
```

```
"Type1Font""CIDFont"
```

```
pdfpostscriptpdfFontspostscriptFonts
```

```
## This duplicates "ComputerModernItalic".
CMitalic <- Type1Font("ComputerModern2",
                     c("CM_regular_10.afm", "CM_boldx_10.afm",
                       "cmti10.afm", "cmbxti10.afm",
                       "CM_symbol_10.afm"),
                     encoding = "TeXtext.enc")

## Not run:
## This could be used by
postscript(family = CMitalic)
## or
postscriptFonts(CMitalic = CMitalic) # once in a session
postscript(family = "CMitalic", encoding = "TeXtext.enc")

## End(Not run)
```

```
windows
```

```
windowswin.graphx11X11win.metafilewin.print
```

```
windows(width, height, pointsize, record, rescale, xpinch, ypinch,
        bg, canvas, gamma, xpos, ypos, buffered, title,
        restoreConsole, clickToConfirm, fillOddEven,
        family, antialias)
```

```
win.graph(width, height, pointsize)
```

```
win.metafile(filename = "", width = 7, height = 7, pointsize = 12,
             family, restoreConsole = TRUE,
             xpinch = NA_real_, ypinch = NA_real_)
```

```
win.print(width = 7, height = 7, pointsize = 12, printer = "",
          family, antialias, restoreConsole = TRUE)
```

```

widthheight      7
pointsize        12
record           FALSE
rescale          c("R", "fit", "fixed")"R"
xpinchypinch     NA_real_
bg               "transparent"
canvas           "white"
gamma
xposypos         Rconsolexpos = -25, ypos = 0
buffered         TRUE
title            ""
filename         .emf.wmfpdfpath.expand""
printer
restoreConsole   FALSE
clickToConfirm   TRUE
fillOddEven      polygonTRUE
family
antialias        "default""none""cleartype""gray"

```

```

windowswindows.optionsgamma xpinchypinchbufferedrestoreConsoleantialiaswin.graph
x11X11
xpinchypinchwindows.options
bgpar("bg")gpar("fill")
"SavedPlots"print"recordedplot"recordPlot.SavedPlots
options("windowsTimeout")
par(lwd =)
win.metafilefilenamefilename""
restoreConsoleFALSE

```

```

"R""fit""fixed"
rescalerescale = "fit"rescale = "fixed"
strwidthstrheight"fit"
"din"

```

```
\etc\Rdevgpar(font =)familypar(family =)windowsFonts  
antialiasantialias = "none""cleartype"
```

Meiryo

```
pch = "."cex = 1
```

```
x11()X11()win.graph()windows()  
x11()X11()
```

```
windowsFontssavePlotbringToTopDevicespdfx11
```

```
## Not run: ## A series of plots written to a sequence of metafiles  
if(.Platform$OS.type == "windows")  
  win.metafile("Rplot%02d.wmf", pointsize = 10)  
  
## End(Not run)
```

windows.options

```
windows.optionswindows  
windows.optionswindowswindows
```

```
windows.options(..., reset = FALSE)
```

```
...      widthheightpointsize-recordrescalexpinchypinchbgcanvasgamma  
         xposyposbufferedrestoreConsoleclickToConfirmtitlefillOddEven  
         antialias
```

```
reset
```

```
reset = TRUE...
antialiaswindowswin.graphX11x11bitmap.aa.wintype = "windows"
```

[windowssps.options](#)

```
## Not run:
## put something like this in your .Rprofile to customize the defaults
setHook(packageEvent("grDevices", "onLoad"),
  function(...)
    grDevices::windows.options(width = 8, height = 6,
                               xpos = 0, pointsize = 10,
                               bitmap.aa.win = "cleartype"))

## End(Not run)
```

windowsFonts

windowsFont(family)

windowsFonts(...)

```
family      "TT"
...
```

windows"family"[par](#)"fontfamily"[gpar](#)

windowsFontswindowsFont
"sans""serif""mono"

[windows](#)

```

if(.Platform$OS.type == "windows") withAutoprint({
  windowsFonts()
  windowsFonts("mono")
})

## Not run: ## set up for Japanese: needs the fonts installed
windows() # make sure we have the right device type (available on Windows only)
Sys.setlocale("LC_ALL", "ja")
windowsFonts(JP1 = windowsFont("MS Mincho"),
             JP2 = windowsFont("MS Gothic"),
             JP3 = windowsFont("Arial Unicode MS"))
plot(1:10)
text(5, 2, "\u{4E10}\u{4E00}\u{4E01}", family = "JP1")
text(7, 2, "\u{4E10}\u{4E00}\u{4E01}", family = "JP1", font = 2)
text(5, 1.5, "\u{4E10}\u{4E00}\u{4E01}", family = "JP2")
text(9, 2, "\u{5100}", family = "JP3")

## End(Not run)

```

x11

```

X11()x11()windows()x11()X11()
X11(*)

```

```

X11
x11X11
https://en.wikipedia.org/wiki/Xlibhttps://www.cairographics.org

```

```

X11(display = "", width, height, pointsize, gamma, bg, canvas,
     fonts, family, xpos, ypos, title, type, antialias, symbolfamily)

```

```

X11.options(..., reset = FALSE)

```

display	DISPLAY
widthheight	NA7
pointsize	12
gamma	
bg	"transparent"
canvas	"white"
fonts	type = "Xlib"
family	type = "Xlib" X11Fonts() fonts
xposypos	xpos = -100NA
title	"" filedf


```
type          "Xlib""cairo""nbcairo""dbcairo""cairo"pangocairo"Xlib"
antialias     c("default", "none", "gray", "subpixel")
symbolfamily
reset
...          X11colortypemaxcubysize
```

```
X11X11.options
```

```
typetype = "Xlib""cairo""nbcairo""dbcairo"
```

```
type = "nbcairo" type = "cairo" dev.hold dev.flush type = "dbcairo" options(X11updates
= 0.25)
x86_64 type = "dbcairo"
type = "cairo"
type = "Xlib" pngdisplay
```

```
type = "Xlib"
fontspar(family =)X11FontsfamilyfontsX11.options
iso10646-1
fonts
"-*-mincho-%s-%s-*-%d-*-*-*-*-*"-*-cronyx-helvetica-%s-%s-*-%d-*-*-*-*-*"
```

```
XLC_LOCALEru_RU.utf8
```

```
"Helvetica"familypargpar"sans""Helvetica""serif""Times""mono""Courier"
PangofontconfigfontconfigFC_DEBUG
https://www.freedesktop.org/software/fontconfig/fontconfig-user.html
"Helvetica""Arial""DejaVu Sans""Liberation Sans""FreeSans""Times New Roman"
"DejaVu Serif""Liberation Serif""CM Roman"
"symbol"symbolfamilyX11.options
quote(pi)expression(10^degree)fontconfig~/.fonts.conf/etc/fonts/local.conf
<fontconfig>
<match target="pattern">
  <test name="family"><string>Symbol</string></test>
  <edit name="family" mode="prepend" binding="same">
    <string>Standard Symbols L</string>
  </edit>
</match>
</fontconfig>
```

```
fc-match Symbolsymbol.ttflocate symbol.ttfwine
```

```
geometryNAX11.optionsR_x11man Xhttps://www.x.org/releases/current/~/.Xresources
```

```
R_x11*geometry: 900x900-0+0
```

```
X11
```

```
type = "Xlib"colortypeX11.optionsX11type = "Xlib"bmpjpegpnggtiffcolortype  
colortypeX11.options(colortype =)  
"true""pseudo""gray""mono""pseudo""pseudo.cube"  
colortype"pseudo.cube""gray""mono""gray""pseudo.cube"X11.options(maxcolorsize  
=)
```

```
type = "Xlib"StaticGrayMonoChrome
```

```
image
```

```
antialias = "default"antialias = "gray"
```

```
type = "Xlib"
```

```
type = "Xlib"
```

```
X11()capabilities("X11")
```

```
DevicesX11FontssavePlot
```

```
## Not run:  
if(.Platform$OS.type == "unix") { # Only on unix-alikes, possibly Mac,  
## put something like this is your .Rprofile to customize the defaults  
setHook(packageEvent("grDevices", "onLoad"),  
        function(...) grDevices::X11.options(width = 8, height = 6, xpos = 0,  
                                                pointsize = 10))  
}  
## End(Not run)
```

X11Fonts

X11Font(font)

X11Fonts(...)

font

...

X11type = "Xlib"X11(type = "cairo")

X11"family"par"fontfamily"gpar

X11FontsX11Font

"sans""serif""mono""Helvetica""Times""CyrHelvetica""CyrTimes""Arial""Mincho"

capabilities()[["X11"]]

X11

```
if(capabilities("X11")) withAutoprint({  
  X11Fonts()  
  X11Fonts("mono")  
  utopia <- X11Font("--utopia*-*-*-*-*-*-*-*-*")  
  X11Fonts(utopia = utopia)  
})
```

xy.coords

xy.coords

xy.coords(x, y = NULL, xlab = NULL, ylab = NULL, log = NULL,
 recycle = FALSE, setLab = TRUE)

```

xy          x
xlabylab
log         "x""y"plotNA"log_le_0"
recycle     TRUErepxy
setLab      xlabylab(x,y)xlabylab

```

```

xy
yNULLx

yvar ~ xvarxvaryvar
xy
time(x)
data.frame x"x""y"
xx1:nxlab"Index"setLab
x"POSIXt""POSIXct"

```

```

x          "double"
y          x
xlab       character(1)NULLx
ylab       character(1)NULLy

```

```

plot.defaultlinespointslowess

```

```

ff <- stats::fft(1:9)
xy.coords(ff)
xy.coords(ff, xlab = "fft") # labels "Re(fft)", "Im(fft)"

with(cars, xy.coords(dist ~ speed, NULL)$xlab ) # = "speed"

xy.coords(1:3, 1:2, recycle = TRUE) # otherwise error "lengths differ"
xy.coords(-2:10, log = "y")
##> xlab: "Index" \\ warning: 3 y values <= 0 omitted ..
op <- options(warn = 2)# ==> warnings would be errors, we suppress the one "we know":
suppressWarnings(xy.coords(-2:10, log = "y"), classes="log_le_0") -> xy
options(op) # revert
stopifnot(is.list(xy), identical (1:13 +0, xy$x),
          identical(c(rep(NA, 3), 1:10 +0), xy$y))

```

xyTable

```
xyTable(x, y = NULL, digits)
```

```
xy          xy.coords(x, y)
digits      signif(*, digits)
```

```
x
y          x
number     number[i](x[i], y[i])
```

```
sunflowerplotxyTable()signif
```

```
xyTable(iris[, 3:4], digits = 6)

## If missing coordinates exist, they are also counted
iris2 <- iris[1:10, 3:4]
iris2[4, 2] <- NA
iris2[c(3, 5), ] <- NA
xyTable(iris2)

## Discretized uncorrelated Gaussian:

xy <- data.frame(x = round(sort(stats::rnorm(100))), y = stats::rnorm(100))
xyTable(xy, digits = 1)
```

`xyz.coords`

```
xyz.coords(x, y = NULL, z = NULL,  
           xlab = NULL, ylab = NULL, zlab = NULL,  
           log = NULL, recycle = FALSE, setLab = TRUE)
```

```
xyz           yzNULLx  
              zvar ~ xvar + yvarxvaryvarzvarxyzdata.framex  
              xyz = NULL  
  
xlabylabzlab  
log          "x""y""z"NA"log_le_0"  
recycle      TRUErepxyz  
setLab       xlabylab(x,y)xlabylab
```

```
x            double  
y            x  
z            x  
xlab         character(1)NULLx  
ylab         character(1)NULLy  
zlab         character(1)NULLz
```

`xy.coords`

```
xyz.coords(data.frame(10*1:9, -4), y = NULL, z = NULL)  
  
xyz.coords(1:5, stats::fft(1:5), z = NULL, xlab = "X", ylab = "Y")  
  
y <- 2 * (x2 <- 10 + (x1 <- 1:10))  
xyz.coords(y ~ x1 + x2, y = NULL, z = NULL)  
  
xyz.coords(data.frame(x = -1:9, y = 2:12, z = 3:13), y = NULL, z = NULL,  
            log = "xy")  
##> Warning message: 2 x values <= 0 omitted ...  
## Suppress this specific warning:  
suppressWarnings(xyz.coords(x = -1:9, y = 2:12, z = 3:13, log = "xy"),  
                  classes = "log_le_0")
```


graphics

graphics-package

```
library(help = "graphics")
```

```
<R-core@r-project.org>
```

abline

```
abline(a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,  
       coef = NULL, untf = FALSE, ...)
```

```
ab  
untf  
h  
v  
coef  
reg      coef  
...      colltylwdxpdlendljoinlmitre
```



```
abline(a, b, ...)
abline(h =, ...)
abline(v =, ...)
abline(coef =, ...)
abline(reg =, ...)
```

```
a
h=v=
coef
regcoef
untfhv
colltylwdparh=v=
xpdpar("xpd")
```

linessegmentspar

```
## Setup up coordinate system (with x == y aspect ratio):
plot(c(-2,3), c(-1,5), type = "n", xlab = "x", ylab = "y", asp = 1)
## the x- and y-axis, and an integer grid
abline(h = 0, v = 0, col = "gray60")
text(1,0, "abline( h = 0 )", col = "gray60", adj = c(0, -.1))
abline(h = -1:5, v = -2:3, col = "lightgray", lty = 3)
abline(a = 1, b = 2, col = 2)
text(1,3, "abline( 1, 2 )", col = 2, adj = c(-.1, -.1))

## Simple Regression Lines:
require(stats)
sale5 <- c(6, 4, 9, 7, 6, 12, 8, 10, 9, 13)
plot(sale5)
abline(lsfrit(1:10, sale5))
abline(lsfrit(1:10, sale5, intercept = FALSE), col = 4) # less fitting

z <- lm(dist ~ speed, data = cars)
plot(cars)
abline(z) # equivalent to abline(reg = z) or
abline(coef = coef(z))

## trivial intercept model
abline(mC <- lm(dist ~ 1, data = cars)) ## the same as
abline(a = coef(mC), b = 0, col = "blue")
```

arrows

```
arrows(x0, y0, x1 = x0, y1 = y0, length = 0.25, angle = 30,  
       code = 2, col = par("fg"), lty = par("lty"),  
       lwd = par("lwd"), ...)
```

```
x0y0  
x1y1  
length  
angle  
code  
colltylwd      NAcol  
...            xpdlendljoinlmitrepar
```

```
i(x0[i], y0[i])(x1[i], y1[i])  
code = 1(x0[i], y0[i])code = 2(x1[i], y1[i])code = 3length = 0  
colltylwd
```

```
x1, y1, x2, y2
```

segments

```
x <- stats::runif(12); y <- stats::rnorm(12)  
i <- order(x, y); x <- x[i]; y <- y[i]  
plot(x,y, main = "arrows(.) and segments(.)")  
## draw arrows from point to point :  
s <- seq(length(x)-1) # one shorter than data  
arrows(x[s], y[s], x[s+1], y[s+1], col = 1:3)  
s <- s[-length(s)]  
segments(x[s], y[s], x[s+2], y[s+2], col = "pink")
```

assocplot

```
assocplot(x, col = c("black", "red"), space = 0.3,  
          main = NULL, xlab = NULL, ylab = NULL)
```

x

col

space

main

xlab x

ylab x

$$\chi^2_{i,j} d_{ij} = (f_{ij} - e_{ij}) / \sqrt{e_{ij} f_{ij} e_{ij} d_{ij}} \sqrt{e_{ij} d_{ij}} = 0 \text{ col col}$$

[assoc](#)

<http://datavis.ca/papers/sugi/sugi17.pdf>

[mosaicplotchisq.test](#)

```
## Aggregate over sex:  
x <- marginSums(HairEyeColor, c(1, 2))  
x  
assocplot(x, main = "Relation between hair and eye color")
```

Axis

```
Axis(x = NULL, at = NULL, ..., side, labels = NULL)
```

```
x  
at  
side  
labels      at  
...         axis
```

```
xxatataxisx
```

```
boxplotcontourcoplotfilled.contourpairsplot.defaultrugstripchartAxis
```

```
"Date""POSIXt"formataxisaxis.POSIXct
```

```
axisAxis()
```

axis

```
axis(side, at = NULL, labels = TRUE, tick = TRUE, line = NA,  
      pos = NA, outer = FALSE, font = NA, lty = "solid",  
      lwd = 1, lwd.ticks = lwd, col = NULL, col.ticks = NULL,  
      hadj = NA, padj = NA, gap.axis = NA, ...)
```

```

side
at          NaNNANULL
labels      as.graphicsAnnotatlabelsTRUE
tick
line        NA
pos         NAline
outer
font        par("font.axis")
lty
lwdlwd.ticks
colcol.ticks  col = NULLpar("fg")col.ticks = NULLcol
hadj        par("adj")
padj        padj = 0padj = 1
            padjpar("las")
gap.axis     NA10.25

            perpendicular <- function(side, las) {
              is.x <- (side %% 2 == 1) # is horizontal x-axis
              ( is.x && (las %in% 2:3)) ||
              (!is.x && (las %in% 1:2))
            }
            gap.axis <- if(perpendicular(side, las)) 0.25 else 1
gap.axisat = ..
...          cex.axiscol.axisfont.axismgpxaxpyaxptcktcillasfgcolxpdpar
            xaxtyaxt
            lablabelsplot.windowlabaxis

atxpd = TRUExpd = NA
at = NULLaxTicks(side)par("xaxp")"yaxp"par("xlog")"ylog"plotaspplot.window
labelsatprint.default(digits = 7)
gap.axislas = 2gap.axis = -1
linepospar("mgp")[3]par("mgp")[2]pos
mgp[2:3]mextcktclex.axiscol.axisfont.axislassrtsrtatlasadjpar

```

Axis

```
axTicksat = NULLpretty
```

```
par
```

```
require(stats) # for rnorm
plot(1:4, rnorm(4), axes = FALSE)
axis(1, 1:4, LETTERS[1:4])
axis(2)
box() #- to make it look "as usual"

plot(1:7, rnorm(7), main = "axis() examples",
     type = "s", xaxt = "n", frame.plot = FALSE, col = "red")
axis(1, 1:7, LETTERS[1:7], col.axis = "blue")
# unusual options:
axis(4, col = "violet", col.axis = "dark violet", lwd = 2)
axis(3, col = "gold", lty = 2, lwd = 0.5)

# one way to have a custom x axis
plot(1:10, xaxt = "n")
axis(1, xaxp = c(2, 9, 7))

## Changing default gap between labels:
plot(0:100, type="n", axes=FALSE, ann=FALSE)
title(quote("axis(1, .., gap.axis = f)," ~~ f >= 0))
axis(2, at = 5*(0:20), las = 1, gap.axis = 1/4)
gaps <- c(4, 2, 1, 1/2, 1/4, 0.1, 0)
chG <- paste0(ifelse(gaps == 1, "default: ", ""),
              "gap.axis=", formatC(gaps))
jj <- seq_along(gaps)
linG <- -2.5*(jj-1)
for(j in jj) {
  isD <- gaps[j] == 1 # is default
  axis (1, at=5*(0:20), gap.axis = gaps[j], padj=-1, line = linG[j],
        col.axis = if(isD) "forest green" else 1, font.axis= 1+isD)
}
mtext(chG, side=1, padj=-1, line = linG -1/2, cex=3/4,
      col = ifelse(gaps == 1, "forest green", "blue3"))
## now shrink the window (in x- and y-direction) and observe the axis labels drawn
```

axis.POSIXct

"POSIXt""Date"

```
axis.POSIXct(side, x, at, format, labels = TRUE, ...)
axis.Date(side, x, at, format, labels = TRUE, ...)
```

```

side          axis
xat
format        strftime
labels        at
...

ataxis.POSIXctaxis.Dateprettyformatpar("lab")
atformatatlabels = FALSE
"POSIXct""tzone"
"LC_TIME"Sys.setlocale

```

Axis

```

with(beaver1, {
  opar <- par(mfrow = c(3,1))
  time <- strptime(paste(1990, day, time %% 100, time %% 100),
                  "%Y %j %H %M")
  plot(time, temp, type = "l") # axis at 6-hour intervals
  # request more ticks
  olab <- par(lab = c(10, 10, 7))
  plot(time, temp, type = "l")
  par(olab)
  # now label every hour on the time axis
  plot(time, temp, type = "l", xaxt = "n")
  r <- as.POSIXct(round(range(time), "hours"))
  axis.POSIXct(1, at = seq(r[1], r[2], by = "hour"), format = "%H")
  par(opar) # reset changed par settings
})

plot(.leap.seconds, seq_along(.leap.seconds), type = "n", yaxt = "n",
     xlab = "leap seconds", ylab = "", bty = "n")
rug(.leap.seconds)
## or as dates
lps <- as.Date(.leap.seconds)
plot(lps, seq_along(.leap.seconds),
     type = "n", yaxt = "n", xlab = "leap seconds",
     ylab = "", bty = "n")
rug(lps)

## 100 random dates in a 10-week period
random.dates <- as.Date("2001/1/1") + 70*sort(stats::runif(100))
plot(random.dates, 1:100)
# or for a better axis labelling

```

```

plot(random.dates, 1:100, xaxt = "n")
axis.Date(1, at = seq(as.Date("2001/1/1"), max(random.dates)+6, "weeks"))
axis.Date(1, at = seq(as.Date("2001/1/1"), max(random.dates)+6, "days"),
  labels = FALSE, tcl = -0.2)

## axis.Date() with various data types:
x <- seq(as.Date("2022-01-20"), as.Date("2023-03-21"), by = "days")
plot(data.frame(x, y = 1), xaxt = "n")
legend("topleft", title = "input",
  legend = c("character", "Date", "POSIXct", "POSIXlt", "numeric"),
  fill = c("violet", "red", "orange", "coral1", "darkgreen"))
axis.Date(1)
axis.Date(3, at = "2022-04-01", col.axis = "violet")
axis.Date(3, at = as.Date("2022-07-01"), col.axis = "red")
axis.Date(3, at = as.POSIXct(as.Date("2022-10-01")), col.axis = "orange")
axis.Date(3, at = as.POSIXlt(as.Date("2023-01-01")), col.axis = "coral1")
axis.Date(3, at = as.integer(as.Date("2023-04-01")), col.axis = "darkgreen")
## automatically extends the format:
axis.Date(1, at = "2022-02-15", col.axis = "violet",
  col = "violet", tck = -0.05, mgp = c(3,2,0))

## axis.POSIXct() with various data types (2 minutes):
x <- as.POSIXct("2022-10-01") + c(0, 60, 120)
attributes(x) # no timezone
plot(data.frame(x, y = 1), xaxt = "n")
legend("topleft", title = "input",
  legend = c("character", "Date", "POSIXct", "POSIXlt", "numeric"),
  fill = c("violet", "red", "orange", "coral1", "darkgreen"))
axis.POSIXct(1)
axis.POSIXct(3, at = "2022-10-01 00:01", col.axis = "violet")
axis.POSIXct(3, at = as.Date("2022-10-01"), col.axis = "red")
axis.POSIXct(3, at = as.POSIXct("2022-10-01 00:01:30"), col.axis = "orange")
axis.POSIXct(3, at = as.POSIXlt("2022-10-01 00:02"), col.axis = "coral1")
axis.POSIXct(3, at = as.numeric(as.POSIXct("2022-10-01 00:00:30")),
  col.axis = "darkgreen")
## automatically extends format (here: subseconds):
axis.POSIXct(3, at = as.numeric(as.POSIXct("2022-10-01 00:00:30")) + 0.25,
  col.axis = "forestgreen", col = "darkgreen", mgp = c(3,2,0))

## axis.POSIXct: 2 time zones
HST <- as.POSIXct("2022-10-01", tz = "HST") + c(0, 60, 60*60)
CET <- HST
attr(CET, "tzzone") <- "CET"
plot(data.frame(HST, y = 1), xaxt = "n", xlab = "Hawaii Standard Time (HST)")
axis.POSIXct(1, HST)
axis.POSIXct(1, HST, at = "2022-10-01 00:10", col.axis = "violet")
axis.POSIXct(3, CET)
mtext(3, text = "Central European Time (CET)", line = 3)
axis.POSIXct(3, CET, at="2022-10-01 12:10", col.axis = "violet")

```

axTicks

ataxis(side)


```
axTicks(side, axp = NULL, usr = NULL, log = NULL, nintLog = NULL)
```

```
side          axis
axp           par("xaxp")par("yaxp")sidepar("xaxp")sidepar("yaxp")
usr           par("usr")par("usr")[1:2]par("usr")[3:4]side
log           par("xlog")par("ylog")side
nintLog       logpar("lab")[j]jsideInf
```

```
axpusrlogpar(..)axplogTRUEpar(xaxp = .)
axTicks()CreateAtVector().../src/main/plot.caxis(side, *)ataxisTicks()
log = TRUEaxisTicksnintLog = Inf
```

```
axis(side)usr
```

```
axisparprettylog = TRUE.
```

```
axisTicks()
```

```
plot(1:7, 10*21:27)
axTicks(1)
axTicks(2)
stopifnot(identical(axTicks(1), axTicks(3)),
           identical(axTicks(2), axTicks(4)))

## Show how axTicks() and axis() correspond :
op <- par(mfrow = c(3, 1))
for(x in 9999 * c(1, 2, 8)) {
  plot(x, 9, log = "x")
  cat(formatC(par("xaxp"), width = 5),",", T <- axTicks(1),"\\n")
  rug(T, col = adjustcolor("red", 0.5), lwd = 4)
}
par(op)

x <- 9.9*10^(-3:10)
plot(x, 1:14, log = "x")
axTicks(1) # now length 7
axTicks(1, nintLog = Inf) # rather too many
```

```
## An example using axTicks() without reference to an existing plot
## (copying R's internal procedures for setting axis ranges etc.),
## You do need to supply _all_ of axp, usr, log, nintLog
## standard logarithmic y axis labels
ylims <- c(0.2, 88)
get_axp <- function(x) 10^c(ceiling(x[1]), floor(x[2]))
## mimic par("yaxs") == "i"
usr.i <- log10(ylims)
```

```

(aT.i <- axTicks(side = 2, usr = usr.i,
                 axp = c(get_axp(usr.i), n = 3), log = TRUE, nintLog = 5))
## mimic (default) par("yaxs") == "r"
usr.r <- extendrange(r = log10(ylims), f = 0.04)
(aT.r <- axTicks(side = 2, usr = usr.r,
                 axp = c(get_axp(usr.r), 3), log = TRUE, nintLog = 5))

## Prove that we got it right :
plot(0:1, ylims, log = "y", yaxs = "i")
stopifnot(all.equal(aT.i, axTicks(side = 2)))

plot(0:1, ylims, log = "y", yaxs = "r")
stopifnot(all.equal(aT.r, axTicks(side = 2)))

```

barplot

```

barplot(height, ...)

## Default S3 method:
barplot(height, width = 1, space = NULL,
        names.arg = NULL, legend.text = NULL, beside = FALSE,
        horiz = FALSE, density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, ann = !add && par("ann"), args.legend = NULL, ...)

## S3 method for class 'formula'
barplot(formula, data, subset, na.action,
        horiz = FALSE, xlab = NULL, ylab = NULL, ...)

```

height	heightheightbesideFALSEheightheightbesideTRUE
width	xlim
space	heightbesideTRUEspacec(0,1)heightbesideTRUE
names.arg	namesheight
legend.text	heightheightlegend.textheight
beside	FALSEheightTRUE
horiz	FALSETRUE
density	NULLdensity
angle	

```

col          "grey"heightheightgrey.colors
border       border = NAborder = TRUE
mainsub
xlab
ylab
xlim
ylim
xpd
log          plot.default
axes        TRUEhoriz
axisnames   TRUEenames.arglty = 0
cex.axis    par('cex')
cex.names
inside      TRUEspace = 0beside = TRUE
plot        FALSE
axis.lty    ltypar('lty')
offset
add         FALSE
ann         mainsubxlabylabtitle
args.legend legend()legend.text
formula     yx
           y ~ x
           y ~ x1 + x2
           cbind(y1, y2) ~ x

data
subset
na.action   NA
...         axesaspmainparplot.window()title()axis

beside = TRUEmp
besidecolMeans(mp)

```

```
plot(..., type = "h")dotcharthistmosaicplot()
```

```
# Formula method
barplot(GNP ~ Year, data = longley)
barplot(cbind(Employed, Unemployed) ~ Year, data = longley)

## 3rd form of formula - 2 categories :
op <- par(mfrow = 2:1, mgp = c(3,1,0)/2, mar = .1+c(3,3:1))
summary(d.Titanic <- as.data.frame(Titanic))
barplot(Freq ~ Class + Survived, data = d.Titanic,
        subset = Age == "Adult" & Sex == "Male",
        main = "barplot(Freq ~ Class + Survived, *)", ylab = "# {passengers}", legend.text = TRUE)
# Corresponding table :
(xt <- xtabs(Freq ~ Survived + Class + Sex, d.Titanic, subset = Age=="Adult"))
# Alternatively, a mosaic plot :
mosaicplot(xt[,,"Male"], main = "mosaicplot(Freq ~ Class + Survived, *)", color=TRUE)
par(op)
```

```
# Default method
require(grDevices) # for colours
tN <- table(Ni <- stats::rpois(100, lambda = 5))
r <- barplot(tN, col = rainbow(20))
#- type = "h" plotting *is* 'bar'plot
lines(r, tN, type = "h", col = "red", lwd = 2)

barplot(tN, space = 1.5, axisnames = FALSE,
        sub = "barplot(..., space= 1.5, axisnames = FALSE)")
```

```
barplot(VADeaths, plot = FALSE)
barplot(VADeaths, plot = FALSE, beside = TRUE)
```

```
mp <- barplot(VADeaths) # default
tot <- colMeans(VADeaths)
text(mp, tot + 3, format(tot), xpd = TRUE, col = "blue")
barplot(VADeaths, beside = TRUE,
        col = c("lightblue", "mistyrose", "lightcyan",
                "lavender", "cornsilk"),
        legend.text = rownames(VADeaths), ylim = c(0, 100))
title(main = "Death Rates in Virginia", font.main = 4)
```

```
hh <- t(VADeaths)[, 5:1]
mybarcol <- "gray20"
mp <- barplot(hh, beside = TRUE,
        col = c("lightblue", "mistyrose",
                "lightcyan", "lavender"),
        legend.text = colnames(VADeaths), ylim = c(0,100),
        main = "Death Rates in Virginia", font.main = 4,
        sub = "Faked upper 2*sigma error bars", col.sub = mybarcol,
        cex.names = 1.5)
segments(mp, hh, mp, hh + 2*sqrt(1000*hh/100), col = mybarcol, lwd = 1.5)
stopifnot(dim(mp) == dim(hh)) # corresponding matrices
mtext(side = 1, at = colMeans(mp), line = -2,
        text = paste("Mean", formatC(colMeans(hh))), col = "red")
```

```

# Bar shading example
barplot(VADeaths, angle = 15+10*1:5, density = 20, col = "black",
        legend.text = rownames(VADeaths))
title(main = list("Death Rates in Virginia", font = 4))

# Border color
barplot(VADeaths, border = "dark blue")

# Log scales (not much sense here)
barplot(tN, col = heat.colors(12), log = "y")
barplot(tN, col = gray.colors(20), log = "xy")

# Legend location
barplot(height = cbind(x = c(465, 91) / 465 * 100,
                        y = c(840, 200) / 840 * 100,
                        z = c(37, 17) / 37 * 100),
        beside = FALSE,
        width = c(465, 840, 37),
        col = c(1, 2),
        legend.text = c("A", "B"),
        args.legend = list(x = "topleft"))

```

box

bty^{par}

box(which = "plot", lty = "solid", ...)

which "plot" "figure" "inner" "outer"

lty

... btycollwd^{par}xpd

colNAfgNApar("col")

^{rect}

```

plot(1:7, abs(stats::rnorm(7)), type = "h", axes = FALSE)
axis(1, at = 1:7, labels = letters[1:7])
box(lty = '1373', col = 'red')

```

boxplot

```
boxplot(x, ...)
```

```
## S3 method for class 'formula'
```

```
boxplot(formula, data = NULL, ..., subset, na.action = NULL,  
        xlab = mklab(y_var = horizontal),  
        ylab = mklab(y_var != horizontal),  
        add = FALSE, ann = !add, horizontal = FALSE,  
        drop = FALSE, sep = ".", lex.order = FALSE)
```

```
## Default S3 method:
```

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,  
        notch = FALSE, outline = TRUE, names, plot = TRUE,  
        border = par("fg"), col = "lightgray", log = "",  
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),  
        ann = !add, horizontal = FALSE, add = FALSE, at = NULL)
```

formula	y ~ grp _{py} grp ~ g ₁ + g ₂ g ₁ :g ₂
---------	--

data	formula
------	---------

subset	
--------	--

na.action	NA
-----------	----

xlaby	lab
ann	FALSE

ann	logical xlaby
-----	-------------------------------

drop	se
plex	order
	split.default

x	NA
---	--------------------

...	formula
	xbxp pars pars bxp

range	range
-------	-------

width	
-------	--

varwidth	varwidthTRUE
----------	--------------

notch	notchTRUE boxplot.stats
-------	---

outline	outline
---------	---------

names	
-------	--

boxwex	
--------	--

staplewex	
-----------	--

outwex	
--------	--

plot	TRUE
------	------

```

border      borderborder
col          col
log
pars         boxwexoutpchbxpplot
horizontal   FALSE
add
at           add = TRUE1:nn

```

```

boxplotboxplot.defaultboxplot.formula
factor

```

```

stats
n          NA
conf
out
group      out
names

```

```

boxplot.stats

```

```

boxplot.statsbxpstripchart

```

```

## boxplot on a formula:
boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
# *add* notches (somewhat funny here <--> warning "notches .. outside hinges"):
boxplot(count ~ spray, data = InsectSprays,
        notch = TRUE, add = TRUE, col = "blue")

boxplot(decrease ~ treatment, data = OrchardSprays, col = "bisque",
        log = "y")
## horizontal=TRUE, switching y <--> x :
boxplot(decrease ~ treatment, data = OrchardSprays, col = "bisque",
        log = "x", horizontal=TRUE)

rb <- boxplot(decrease ~ treatment, data = OrchardSprays, col = "bisque")
title("Comparing boxplot()s and non-robust mean +/- SD")
mn.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, mean)

```

```

sd.t <- tapply(OrchardSprays$decrease, OrchardSprays$treatment, sd)
xi <- 0.3 + seq(rb$N)
points(xi, mn.t, col = "orange", pch = 18)
arrows(xi, mn.t - sd.t, xi, mn.t + sd.t,
       code = 3, col = "pink", angle = 75, length = .1)

## boxplot on a matrix:
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
            `5T` = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
boxplot(mat) # directly, calling boxplot.matrix()

## boxplot on a data frame:
df. <- as.data.frame(mat)
par(las = 1) # all axis labels horizontal
boxplot(df., main = "boxplot(*, horizontal = TRUE)", horizontal = TRUE)

## Using 'at = ' and adding boxplots -- example idea by Roger Bivand :
boxplot(len ~ dose, data = ToothGrowth,
        boxwex = 0.25, at = 1:3 - 0.2,
        subset = supp == "VC", col = "yellow",
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg",
        ylab = "tooth length",
        xlim = c(0.5, 3.5), ylim = c(0, 35), yaxs = "i")
boxplot(len ~ dose, data = ToothGrowth, add = TRUE,
        boxwex = 0.25, at = 1:3 + 0.2,
        subset = supp == "OJ", col = "orange")
legend(2, 9, c("Ascorbic acid", "Orange juice"),
      fill = c("yellow", "orange"))

## With less effort (slightly different) using factor *interaction*:
boxplot(len ~ dose:supp, data = ToothGrowth,
        boxwex = 0.5, col = c("orange", "yellow"),
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg", ylab = "tooth length",
        sep = ":", lex.order = TRUE, ylim = c(0, 35), yaxs = "i")

## more examples in help(bxp)

```

boxplot.matrix

```

## S3 method for class 'matrix'
boxplot(x, use.cols = TRUE, ...)

```

```

x
use.cols      use.cols = FALSE
...           boxplot

```


[boxplot](#)

[boxplot.default](#)[boxplot.formula](#)[boxplot.factor](#)

```
## Very similar to the example in ?boxplot
mat <- cbind(Uni05 = (1:100)/21, Norm = rnorm(100),
             T5 = rt(100, df = 5), Gam2 = rgamma(100, shape = 2))
boxplot(mat, main = "boxplot.matrix(..., main = ...)",
        notch = TRUE, col = 1:4)
```

[bxp](#)

[bxp](#)[boxplot](#)

```
bxp(z, notch = FALSE, width = NULL, varwidth = FALSE,
    outline = TRUE, notch.frac = 0.5, log = "",
    border = par("fg"), pars = NULL, frame.plot = axes,
    horizontal = FALSE, ann = TRUE,
    add = FALSE, at = NULL, show.names = NULL,
    ...)
```

<code>z</code>	boxplot
<code>notch</code>	<code>notchTRUE</code>
<code>width</code>	
<code>varwidth</code>	<code>varwidthTRUE</code>
<code>outline</code>	<code>outline</code>
<code>notch.frac</code>	<code>notch = TRUE</code>
<code>border</code>	<code>boxcolmedcolwhiskcolstaplecoloutcol</code>
<code>log</code>	plot.default
<code>frame.plot</code>	box <code>TRUEaxes = FALSE</code>
<code>horizontal</code>	<code>FALSE</code>
<code>ann</code>	
<code>add</code>	
<code>at</code>	<code>add = TRUE1:nn</code>
<code>show.names</code>	<code>TRUEFALSE</code>

```

pars...      pars.....pars
              yaxsyylimhorizontalxlimxaxtyaxtlascecx.axisgap.axiscol.axisaxis
              maincecx.maincol.mainsubcecx.subcol.subxlabylabcecx.labcol.lab
              title
              axesplot.windowTRUE
              parsboxltyltyparspar
              at0.8

              colpar("bg")
              medpch = NAmedlty = "blank"medlwd3×lwd
              "dashed"

              outlty = "blank"outpch = NA

```

at

```

add = FALSExlimxlim = range(at, *) + c(-0.5, 0.5)xlimwidth

```

```

require(stats)
set.seed(753)
(bx.p <- boxplot(split(rt(100, 4), gl(5, 20))))
op <- par(mfrow = c(2, 2))
bxp(bx.p, xaxt = "n")
bxp(bx.p, notch = TRUE, axes = FALSE, pch = 4, boxfill = 1:5)
bxp(bx.p, notch = TRUE, boxfill = "lightblue", frame.plot = FALSE,
    outline = FALSE, main = "bxp(*, frame.plot= FALSE, outline= FALSE)")
bxp(bx.p, notch = TRUE, boxfill = "lightblue", border = 2:6,
    ylim = c(-4,4), pch = 22, bg = "green", log = "x",
    main = "... log = 'x', ylim = *")
par(op)
op <- par(mfrow = c(1, 2))

## single group -- no label
boxplot(weight ~ group, data = PlantGrowth, subset = group == "ctrl")
## with label
bx <- boxplot(weight ~ group, data = PlantGrowth,
    subset = group == "ctrl", plot = FALSE)
bxp(bx, show.names=TRUE)
par(op)

## passing gap.axis=* to axis(), PR#18109:
boxplot(matrix(100*rnorm(1e3), 50, 20),
    cex.axis = 1.5, gap.axis = -1)# showing *all* labels

```

```

z <- split(rnorm(1000), rpois(1000, 2.2))
boxplot(z, whisklty = 3, main = "boxplot(z, whisklty = 3)")

## Colour support similar to plot.default:
op <- par(mfrow = 1:2, bg = "light gray", fg = "midnight blue")
boxplot(z, col.axis = "skyblue3", main = "boxplot(*, col.axis=..,main=..)")
plot(z[[1]], col.axis = "skyblue3", main = "plot(*, col.axis=..,main=..)")
mtext("par(bg=\"light gray\", fg=\"midnight blue\")",
      outer = TRUE, line = -1.2)
par(op)

## Mimic S-Plus:
splus <- list(boxwex = 0.4, staplewex = 1, outwex = 1, boxfill = "grey40",
             medlwd = 3, medcol = "white", whisklty = 3, outlty = 1, outpch = NA)
boxplot(z, pars = splus)
## Recycled and "sweeping" parameters
op <- par(mfrow = c(1,2))
boxplot(z, border = 1:5, lty = 3, medlty = 1, medlwd = 2.5)
boxplot(z, boxfill = 1:3, pch = 1:5, lwd = 1.5, medcol = "white")
par(op)
## too many possibilities
boxplot(z, boxfill = "light gray", outpch = 21:25, outlty = 2,
      bg = "pink", lwd = 2,
      medcol = "dark blue", medcex = 2, medpch = 20)

```

cdplot

yx

cdplot(x, ...)

Default S3 method:

```

cdplot(x, y,
       plot = TRUE, tol.ylab = 0.05, ylevels = NULL,
       bw = "nrd0", n = 512, from = NULL, to = NULL,
       col = NULL, border = 1, main = "", xlab = NULL, ylab = NULL,
       yaxlabels = NULL, xlim = NULL, ylim = c(0, 1), weights = NULL, ...)

```

S3 method for class 'formula'

```

cdplot(formula, data = list(),
       plot = TRUE, tol.ylab = 0.05, ylevels = NULL,
       bw = "nrd0", n = 512, from = NULL, to = NULL,
       col = NULL, border = 1, main = "", xlab = NULL, ylab = NULL,
       yaxlabels = NULL, xlim = NULL, ylim = c(0, 1), ...,
       subset = NULL, weights = NULL)

```

x

y "factor"

```
formula      "formula"y ~ x"factor"
data
plot
tol.ylab
ylevels
bwnfromto... density
col          levels(y)gray.colors
border
mainxlabylab
yaxlabels    levels(y)
xlimylim
subset
weights      NULL
```

```
cdplotxyyy
spineplot $P(y|x)$ density
xx
```

y

<Achim.Zeileis@R-project.org>

spineplotdensity

```
## NASA space shuttle o-ring failures
fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1,
                1, 2, 1, 1, 1, 1, 1),
              levels = 1:2, labels = c("no", "yes"))
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68, 69, 70, 70,
                70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81)

## CD plot
cdplot(fail ~ temperature)
cdplot(fail ~ temperature, bw = 2)
cdplot(fail ~ temperature, bw = "SJ")

## compare with spinogram
(spineplot(fail ~ temperature, breaks = 3))

## highlighting for failures
```

```
cdplot(fail ~ temperature, ylevels = 2:1)

## scatter plot with conditional density
cdens <- cdplot(fail ~ temperature, plot = FALSE)
plot(I(as.numeric(fail) - 1) ~ jitter(temperature, factor = 2),
     xlab = "Temperature", ylab = "Conditional failure probability")
lines(53:81, 1 - cdens[[1]](53:81), col = 2)
```

clip

```
clip(x1, x2, y1, y2)
```

```
x1x2y1y2
```

```
par("xpd")
```

```
plot.newlinestextpar("xpd")boxmtexttitleplot.dendrogramxpd
```

```
par
```

```
x <- rnorm(1000)
hist(x, xlim = c(-4,4))
usr <- par("usr")
clip(usr[1], -2, usr[3], usr[4])
hist(x, col = 'red', add = TRUE)
clip(2, usr[2], usr[3], usr[4])
hist(x, col = 'blue', add = TRUE)
do.call("clip", as.list(usr)) # reset to plot region
```

contour

```
contour(x, ...)
```

```
## Default S3 method:
```

```
contour(x = seq(0, 1, length.out = nrow(z)),
        y = seq(0, 1, length.out = ncol(z)),
        z,
        nlevels = 10, levels = pretty(zlim, nlevels),
        labels = NULL,
        xlim = range(x, finite = TRUE),
        ylim = range(y, finite = TRUE),
        zlim = range(z, finite = TRUE),
        labcex = 0.6, drawlabels = TRUE, method = "flatteest",
        vfont, axes = TRUE, frame.plot = axes,
        col = par("fg"), lty = par("lty"), lwd = par("lwd"),
        add = FALSE, ...)
```

xy	zxlistx\$xx\$yxyzz
z	NAxz
nlevels	levels
levels	
labels	NULLas.character
labcex	cexpar("cex")
drawlabels	TRUE
method	"simple""edge""flatteest"
vfont	NULLtextNULLc("sans serif", "plain")
xlimylimzlim	
axesframe.plot	plot.default
col	
lty	
lwd	
add	TRUE
...	plot.windowtitleAxisboxcex.axis

```
contour
"simple""edge""flatteest"
textHershey
contourzf(x[i], y[j])
> 1colltlylwdlevels
contourplotformulaxyz
collwdlty
```

```

options("max.contour.segments")
contourLinesfilled.contourcontourplotlevelplotimage demo(graphics)

require(grDevices) # for colours
x <- -6:16
op <- par(mfrow = c(2, 2))
contour(outer(x, x), method = "edge", vfont = c("sans serif", "plain"))
z <- outer(x, sqrt(abs(x)), FUN = `^`)
image(x, x, z)
contour(x, x, z, col = "pink", add = TRUE, method = "edge",
        vfont = c("sans serif", "plain"))
contour(x, x, z, ylim = c(1, 6), method = "simple", labcex = 1,
        xlab = quote(x[1]), ylab = quote(x[2]))
contour(x, x, z, ylim = c(-6, 6), nlevels = 20, lty = 2, method = "simple",
        main = "20 levels; \"simple\" labelling method")
par(op)

## Passing multiple colours / lty / lwd :
op <- par(mfrow = c(1, 2))
z <- outer(-9:25, -9:25)
## Using default levels <- pretty(range(z, finite = TRUE), 10),
## the first and last of which typically are *not* drawn:
(levs <- pretty(z, n=10)) # -300 -200 ... 600 700
contour(z, col = 1:4)
## Set levels explicitly; show that 'lwd' and 'lty' are recycled as well:
contour(z, levels=levs[-c(1,length(levs))], col = 1:5, lwd = 1:3 *1.5, lty = 1:3)
par(op)

## Persian Rug Art:
x <- y <- seq(-4*pi, 4*pi, length.out = 27)
r <- sqrt(outer(x^2, y^2, `+`))
opar <- par(mfrow = c(2, 2), mar = rep(0, 4))
for(f in pi^(0:3))
  contour(cos(r^2)*exp(-r/f),
          drawlabels = FALSE, axes = FALSE, frame.plot = TRUE)

rx <- range(x <- 10*1:nrow(volcano))
ry <- range(y <- 10*1:ncol(volcano))
ry <- ry + c(-1, 1) * (diff(rx) - diff(ry))/2
tcol <- terrain.colors(12)
par(opar); opar <- par(pty = "s", bg = "lightcyan")
plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
u <- par("usr")
rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
contour(x, y, volcano, col = tcol[2], lty = "solid", add = TRUE,
        vfont = c("sans serif", "plain"))
title("A Topographic Map of Maunga Whau", font = 4)
abline(h = 200*0:4, v = 200*0:4, col = "lightgray", lty = 2, lwd = 0.1)

## contourLines produces the same contour lines as contour

```

```

plot(x = 0, y = 0, type = "n", xlim = rx, ylim = ry, xlab = "", ylab = "")
u <- par("usr")
rect(u[1], u[3], u[2], u[4], col = tcol[8], border = "red")
contour(x, y, volcano, col = tcol[1], lty = "solid", add = TRUE,
        vfont = c("sans serif", "plain"))
line.list <- contourLines(x, y, volcano)
invisible(lapply(line.list, lines, lwd=3, col=adjustcolor(2, .3)))
par(opar)

```

convertXY

```

grconvertX(x, from = "user", to = "user")
grconvertY(y, from = "user", to = "user")

```

xy
fromto

"user"
 "inches"
 "device"
 "ndc"
 "nfc"
 "npc"
 "nic"
 "lines" mex
 "chars" cex

[pdfpostscript](#)

```

op <- par(omd=c(0.1, 0.9, 0.1, 0.9), mfrow = c(1, 2))
plot(1:4)
for(tp in c("in", "dev", "ndc", "nfc", "npc", "nic", "lines", "chars"))
  print(grconvertX(c(1.0, 4.0), "user", tp))
par(op)

```

coplot

```
coplot(formula, data, given.values, panel = points, rows, columns,
       show.given = TRUE, col = par("fg"), pch = par("pch"),
       bar.bg = c(num = gray(0.8), fac = gray(0.95)),
       xlab = c(x.name, paste("Given :", a.name)),
       ylab = c(y.name, paste("Given :", b.name)),
       subscripts = FALSE,
       axlabels = function(f) abbreviate(levels(f)),
       number = 6, overlap = 0.5, xlim, ylim, ...)
co.intervals(x, number = 6, overlap = 0.5)
```

formula	$y \sim x \mid ayxay \sim x \mid a * byxab$ <code>xyas.numeric()</code> <code>abshow.given</code>
data	<code>coplot</code>
given.values	<code>ab</code> <code>babco.intervalsgiven.values</code>
panel	<code>function(x, y, col, pch, ...)</code> <code>points</code>
rows	<code>rowscolumnsrows</code>
columns	
show.given	<code>TRUE</code>
col	
pch	
bar.bg	<code>"num"</code> <code>"fac"</code>
xlab	
ylab	
subscripts	<code>subscripts</code>
axlabels	
number	<code>factor</code>
overlap	
xlim	
ylim	
...	
x	

```
arowscolumnscolumns >= rows
rowsa
plotboxplot
xlabylabparcex.labfont.labmtexttitle
```

```
co.intervals(., number, .)number×matrixcici[k,]rangexk
```

[pairspanel.smoothpoints](#)

```
## Tonga Trench Earthquakes
coplot(lat ~ long | depth, data = quakes)
given.depth <- co.intervals(quakes$depth, number = 4, overlap = .1)
coplot(lat ~ long | depth, data = quakes, given.values = given.depth, rows = 1)

## Conditioning on 2 variables:
ll.dm <- lat ~ long | depth * mag
coplot(ll.dm, data = quakes)
coplot(ll.dm, data = quakes, number = c(4, 7), show.given = c(TRUE, FALSE))
coplot(ll.dm, data = quakes, number = c(3, 7),
        overlap = c(-.5, .1)) # negative overlap DROPS values

## given two factors
Index <- seq_len(nrow(warpbreaks)) # to get nicer default labels
coplot(breaks ~ Index | wool * tension, data = warpbreaks,
        show.given = 0:1)
coplot(breaks ~ Index | wool * tension, data = warpbreaks,
        col = "red", bg = "pink", pch = 21,
        bar.bg = c(fac = "light blue"))

## Example with empty panels:
with(data.frame(state.x77), {
  coplot(Life.Exp ~ Income | Illiteracy * state.region, number = 3,
        panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...))
  ## y ~ factor -- not really sensible, but 'show off':
  coplot(Life.Exp ~ state.region | Income * state.division,
        panel = panel.smooth)
})
```

curve

[\[from, to\]curvexname](#)

```
curve(expr, from = NULL, to = NULL, n = 101, add = FALSE,
      type = "l", xname = "x", xlab = xname, ylab = NULL,
      log = NULL, xlim = NULL, ...)

## S3 method for class 'function'
plot(x, y = 0, to = 1, from = y, xlim = NULL, ylab = NULL, ...)
```

```

expr          xx
x
y             fromplot
fromto
n
add           TRUENAFALSE
xlim          NULLNULLc(from, to)add = TRUEplot.window
type          plot.default
xname
xlabylablog... log
              "function"plot...curveexpr

```

```

exprcurveplotn[from, to]
fromtoNULLxlimNULL
fromtolimplot(<function>)curve(add = FALSE)(0,1)curve(add = NA)curve(add = TRUE)
logadd = TRUE
logadd = TRUEadd = NAadd = FALSE""

```

```

expr
curveexprprepreprepreprxnameexprxnameall.varsexprxnamenncurve(x, ...)xfunction
plot
plotplot.function

```

```
xy
```

```
add"function"plotaddcurve
```

```
splinefunlines
```

```

plot(qnorm) # default range c(0, 1) is appropriate here,
             # but end values are -/+Inf and so are omitted.
plot(qlogis, main = "The Inverse Logit : qlogis()")
abline(h = 0, v = 0:2/2, lty = 3, col = "gray")

curve(sin, -2*pi, 2*pi, xname = "t")
curve(tan, xname = "t", add = NA,
      main = "curve(tan) --> same x-scale as previous plot")

op <- par(mfrow = c(2, 2))
curve(x^3 - 3*x, -2, 2)

```

```

curve(x^2 - 2, add = TRUE, col = "violet")

## simple and advanced versions, quite similar:
plot(cos, -pi, 3*pi)
curve(cos, xlim = c(-pi, 3*pi), n = 1001, col = "blue", add = TRUE)

chippy <- function(x) sin(cos(x)*exp(-x/2))
curve(chippy, -8, 7, n = 2001)
plot (chippy, -8, -5)

for(ll in c("", "x", "y", "xy"))
  curve(log(1+x), 1, 100, log = ll, sub = paste0("log = '", ll, "'"))
par(op)

```

dotchart

```

dotchart(x, labels = NULL, groups = NULL, gdata = NULL, offset = 1/8,
  ann = par("ann"), xaxt = par("xaxt"), frame.plot = TRUE, log = "",
  cex = par("cex"), pt.cex = cex,
  pch = 21, gpch = 21, bg = par("bg"),
  color = par("fg"), gcolor = par("fg"), lcolor = "gray",
  xlim = range(x[is.finite(x)]),
  main = NULL, xlab = NULL, ylab = NULL, ...)

```

x	<code>NA</code> axis.numeric(x) <code>is.vector(x) is.matrix(x)</code> as.numeric
labels	<code>names(x)</code> <code>dimnames(x)[[1]]</code>
groups	<code>xxgroupsx</code>
gdata	
offset	<code>ylablabels</code>
ann	logical
xaxt	<code>"n"</code> par("xaxt")
frame.plot	
log	plot.default
cex	<code>cexpar("cex")</code>
pt.cex	<code>cexcexplot()</code>
pch	
gpch	
bg	par(bg= *)
color	
gcolor	
lcolor	
xlim	plot.window
main	title
xlaby	<code>title</code>
lab	
...	

```

dotchart(VADeaths, main = "Death Rates in Virginia - 1940")

op <- par(xaxs = "i") # 0 -- 100%
dotchart(t(VADeaths), xlim = c(0,100), bg = "skyblue",
         main = "Death Rates in Virginia - 1940", xlab = "rate [ % ]",
         ylab = "Grouping: Age x Urbanity . Gender")
par(op)

```

filled.contour

```

filled.contour(x = seq(0, 1, length.out = nrow(z)),
               y = seq(0, 1, length.out = ncol(z)),
               z,
               xlim = range(x, finite = TRUE),
               ylim = range(y, finite = TRUE),
               zlim = range(z, finite = TRUE),
               levels = pretty(zlim, nlevels), nlevels = 20,
               color.palette = function(n) hcl.colors(n, "YlOrRd", rev = TRUE),
               col = color.palette(length(levels) - 1),
               plot.title, plot.axes, key.title, key.axes, key.border = NULL,
               asp = NA, xaxs = "i", yaxs = "i", las = 1,
               axes = TRUE, frame.plot = axes, ...)

```

```

.filled.contour(x, y, z, levels, col)

```

xy	z.filled.contourxlistx\$xx\$yxyz
z	xz
xlim	
ylim	
zlim	
levels	zz
nlevels	levelsz

```

color.palette
col
plot.title
plot.axes      box
key.title
key.axes
key.border     rect()
asp            y/xplot.window
xaxs
yaxs
las
axesframe.plot plot.default
...           title()

```

NANANANA

NA

.filled.contour

filled.contourlayout

filled.contourplot.axes

[contourimagehcl.colorsgray.colorspalettecontourplotlevelplot](#)

```

require("grDevices") # for colours
filled.contour(volcano, asp = 1) # simple

x <- 10*1:nrow(volcano)
y <- 10*1:ncol(volcano)
filled.contour(x, y, volcano,
  color.palette = function(n) hcl.colors(n, "terrain"),
  plot.title = title(main = "The Topography of Maunga Whau",
    xlab = "Meters North", ylab = "Meters West"),
  plot.axes = { axis(1, seq(100, 800, by = 100))
    axis(2, seq(100, 600, by = 100)) },
  key.title = title(main = "Height\n(meters)"),
  key.axes = axis(4, seq(90, 190, by = 10))) # maybe also asp = 1
mtext(paste("filled.contour(.) from", R.version.string),
  side = 1, line = 4, adj = 1, cex = .66)

```

```

# Annotating a filled contour plot
a <- expand.grid(1:20, 1:20)
b <- matrix(a[,1] + a[,2], 20)
filled.contour(x = 1:20, y = 1:20, z = b,
               plot.axes = { axis(1); axis(2); points(10, 10) })

## Persian Rug Art:
x <- y <- seq(-4*pi, 4*pi, length.out = 27)
r <- sqrt(outer(x^2, y^2, `+`))
## "minimal"
filled.contour(cos(r^2)*exp(-r/(2*pi)), axes = FALSE, key.border=NA)
## rather, the key *should* be labeled (but axes still not):
filled.contour(cos(r^2)*exp(-r/(2*pi)), frame.plot = FALSE,
               plot.axes = {})

```

fourfoldplot

k

```

fourfoldplot(x, color = c("#99CCFF", "#6699CC"),
             conf.level = 0.95,
             std = c("margins", "ind.max", "all.max"),
             margin = c(1, 2), space = 0.2, main = NULL,
             mfrow = NULL, mfcoll = NULL)

```

<i>x</i>	<i>kk</i>
color	
conf.level	
std	"margins""ind.max""all.max""margins"margin"ind.max""all.max"
margin	12c(1, 2)std"margins"
space	
main	
mfrow	c(nr, nc)nrnc
mfcoll	c(nr, nc)nrnc

k

$f_{ij} \sqrt{f_{ij}}$

k

k<http://datavis.ca/papers/4fold/4fold.pdf>

[mosaicplot](#)

```
## Use the Berkeley admission data as in Friendly (1995).
x <- aperm(UCBAdmissions, c(2, 1, 3))
dimnames(x)[[2]] <- c("Yes", "No")
names(dimnames(x)) <- c("Sex", "Admit?", "Department")
stats::ftable(x)

## Fourfold display of data aggregated over departments, with
## frequencies standardized to equate the margins for admission
## and sex.
## Figure 1 in Friendly (1994).
fourfoldplot(marginSums(x, c(1, 2)))

## Fourfold display of x, with frequencies in each table
## standardized to equate the margins for admission and sex.
## Figure 2 in Friendly (1994).
fourfoldplot(x)

## Fourfold display of x, with frequencies in each table
## standardized to equate the margins for admission. but not
## for sex.
## Figure 3 in Friendly (1994).
fourfoldplot(x, margin = 2)
```

[frame](#)

[frameplot.new](#)

[plot.new\(\)](#)
[frame\(\)](#)

[par\("bg"\)X11windowsquartz](#)
"before.plot.new""plot.new"[setHookget](#)

[frame](#)

[plot.windowplot.default](#)

grid

gridnxny

```
grid(nx = NULL, ny = nx, col = "lightgray", lty = "dotted",  
     lwd = par("lwd"), equilogs = TRUE, nintLog = NULL)
```

```
nxny          NULLaxTicksNA  
col  
lty  
lwd  
equilogs      equilogs = FALSE  
nintLog       NULLaxTicks()nxny
```

"atx""aty"

[abline](#)(h = ., v = .)

[plotablinelinespoints](#)

```
plot(1:3)  
grid(NA, 5, lwd = 2) # grid only in y-direction  
  
## maybe change the desired number of tick marks: par(lab = c(mx, my, 7))  
op <- par(mfcol = 1:2)  
with(iris,  
     {  
       plot(Sepal.Length, Sepal.Width, col = as.integer(Species),  
            xlim = c(4, 8), ylim = c(2, 4.5), panel.first = grid(),  
            main = "with(iris, plot(..., panel.first = grid(), ..) )")  
       plot(Sepal.Length, Sepal.Width, col = as.integer(Species),  
            panel.first = grid(3, lty = 1, lwd = 2),  
            main = "... panel.first = grid(3, lty = 1, lwd = 2), ..")  
     }  
)  
par(op)  
  
plot(1:64)  
gr <- grid() # now *invisibly* returns the grid "at" locations
```

```

str(gr)
stopifnot(length(gr) == 2, identical(gr[[1]], gr[[2]]),
           gr[["atx"]] == 10*(0:6))

## In log-scale plots :
plot(8:270, log="xy") ; grid() # at (1, 10, 100); if preferring "all" grid lines:
plot(8:270, log="xy") ; grid(equilog = FALSE) -> grll
stopifnot(identical(grll, list(atx = c(1, 2, 5, 10, 20, 50, 100, 200),
                                aty = c(10, 20, 50, 100, 200))))

x <- 2^(-9:70)
plot(log(x) ~ x, log="xy")
grll <- grid(equilog = FALSE, col = adjustcolor("green", 1/3))
gr20 <- grid(nintLog = 20)
gr25 <- grid(nintLog = 25, col="thistle")
str(gr25)
stopifnot(exprs = {
  grll$aty == c(1, 2, 5, 10, 20, 50)
  length(gr20$atx) >= 20 # 24 effectively
  all.equal(10^(-3:22), gr25$atx, tol = 1e-15) # even tol = 0
})

```

hist

histplot = TRUE"histogram"plot.histogram

hist(x, ...)

```

## Default S3 method:
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE, fuzz = 1e-7,
     density = NULL, angle = 45, col = "lightgray", border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, warn.unused = TRUE, ...)

```

x

breaks

pretty1e6breaksx

```

freq          TRUEcountsFALSEdensityTRUEbreaksprobability
probability   !freq
include.lowest TRUEx[i]breaksright = FALSEbreaks
right         TRUE
fuzz          x[.]breakfuzz
density       NULLdensity
angle
col
border
mainxlabylab  title()ylab"Frequency"freq
xlimylim     xlimplot = TRUE
axes         TRUE
plot         TRUEplot = TRUE
labels       FALSEplot.histogram
nclass       nclassbreaks
warn.unused   plot = FALSEwarn.unused = TRUEhist.default()
...          plot.histogramtitleaxisplot = TRUE

```

breaks

```

right = TRUE(a,b]include.lowestTRUE
right = FALSE[a,b)include.lowest
10-7breaksdensity
breaks"Sturges"nclass.Sturges"Scott""FD""Freedman-Diaconis"nclass.scott
nclass.FDx

```

"histogram"

```

breaks      n + 1breaks
counts      nx[]
density       $\hat{f}(x_i)$ all(diff(breaks) == 1)counts/n $\sum_i \hat{f}(x_i)(b_{i+1} - b_i) = 1b_i$ 
              breaks[i]
mids         n
xname        x
equidist     breaks

```

```
nclass.Sturgesstemdensitytruehist
```

```
barplotplot(*, type = "h")
```

```
op <- par(mfrow = c(2, 2))
hist(islands)
utils::str(hist(islands, col = "gray", labels = TRUE))

hist(sqrt(islands), breaks = 12, col = "lightblue", border = "pink")
##-- For non-equidistant breaks, counts should NOT be graphed unscaled:
r <- hist(sqrt(islands), breaks = c(4*0:5, 10*3:5, 70, 100, 140),
          col = "blue1")
text(r$mids, r$density, r$counts, adj = c(.5, -.5), col = "blue3")
sapply(r[2:3], sum)
sum(r$density * diff(r$breaks)) # == 1
lines(r, lty = 3, border = "purple") # -> lines.histogram(*)
par(op)

require(utils) # for str
str(hist(islands, breaks = 12, plot = FALSE)) #-> 10 (~= 12) breaks
str(hist(islands, breaks = c(12,20,36,80,200,1000,17000), plot = FALSE))

hist(islands, breaks = c(12,20,36,80,200,1000,17000), freq = TRUE,
      main = "WRONG histogram") # and warning

## Extreme outliers; the "FD" rule would take very large number of 'breaks':
XXL <- c(1:9, c(-1,1)*1e300)
hh <- hist(XXL, "FD") # did not work in R <= 3.4.1; now gives warning
## pretty() determines how many counts are used (platform dependently!):
length(hh$breaks) ## typically 1 million -- though 1e6 was "a suggestion only"

## R >= 4.2.0: no "*.5" labels on y-axis:
hist(c(2,3,3,5,5,6,6,6,7))

require(stats)
set.seed(14)
x <- rchisq(100, df = 4)

## Histogram with custom x-axis:
hist(x, xaxt = "n")
axis(1, at = 0:17)

## Comparing data with a model distribution should be done with qqplot()!
qqplot(x, qchisq(ppoints(x), df = 4)); abline(0, 1, col = 2, lty = 2)

## if you really insist on using hist() ... :
hist(x, freq = FALSE, ylim = c(0, 0.2))
curve(dchisq(x, df = 4), col = 2, lty = 2, lwd = 2, add = TRUE)
```

hist.POSIXt

[hist](#)"Date""POSIXt"

```
## S3 method for class 'POSIXt'
hist(x, breaks, ...,
      xlab = deparse1(substitute(x)),
      plot = TRUE, freq = FALSE,
      start.on.monday = TRUE, format, right = TRUE)
```

```
## S3 method for class 'Date'
hist(x, breaks, ...,
      xlab = deparse1(substitute(x)),
      plot = TRUE, freq = FALSE,
      start.on.monday = TRUE, format, right = TRUE)
```

```
x                "POSIXt""Date"
breaks           x"days""weeks""months""quarters""years""secs""mins""hours"
...              hist.defaultinclude.lowestdensitylabels
xlab
plot             TRUE
freq             TRUEcountsFALSE
start.on.monday
                 breaks = "weeks"
format           strptime
right            TRUE
```

```
breaks
breaks = "quarters"min(x)
right = TRUEbreaks"months""quarters""years"right = FALSE
```

"histogram"[hist](#)

[seq.POSIXt](#)[axis.POSIXt](#)[hist](#)

```

hist(.leap.seconds, "years", freq = TRUE)
brks <- seq(ISOdate(1970, 1, 1), ISOdate(2030, 1, 1), "5 years")
hist(.leap.seconds, brks)
rug(.leap.seconds, lwd=2)
## show that 'include.lowest' "works"
stopifnot(identical(c(2L, rep(1L,11)),
  hist(brks, brks, plot=FALSE, include.lowest=TRUE )$counts))
tools::assertError(verbose=TRUE, ##--> 'breaks' do not span range of 'x'
  hist(brks, brks, plot=FALSE, include.lowest=FALSE))
## The default fuzz in hist.default() "kills" this, with a "wrong" message:
try ( hist(brks[-13] + 1, brks, include.lowest = FALSE) )
## and decreasing 'fuzz' solves the issue:
hb <- hist(brks[-13] + 1, brks, include.lowest = FALSE, fuzz = 1e-10)
stopifnot(hb$counts == 1)

## 100 random dates in a 10-week period
random.dates <- as.Date("2001/1/1") + 70*stats::runif(100)
hist(random.dates, "weeks", format = "%d %b")

```

identify

identifyxy

identify(x, ...)

```

## Default S3 method:
identify(x, y = NULL, labels = seq_along(x), pos = FALSE,
  n = length(x), plot = TRUE, atpen = FALSE, offset = 0.5,
  tolerance = 0.25, order = FALSE, ...)

```

xy	xy.coords xy
labels	as.character x
pos	posTRUE
n	
plot	plotTRUEFALSE
atpen	TRUEplot = TRUE
offset	atpen = TRUE
tolerance	
order	orderTRUE
...	par cexcolfont

```

identify
identifyX11windowsquartz
tolerance
plotTRUElabelsatpenatpen
X11quartzCtrlESC
identifyoptions(locatorBell = FALSE)
identifyidentify
identify

posorderFALSE
posorderTRUEindposTRUEposatpen = TRUEorderTRUEorder

```

```

textposposidentify
offsetoffsetoffsetplot = FALSEtextpoints

```

```

locatortext
dev.capabilities

```

```

## A function to use identify to select points, and overplot the
## points with another symbol as they are selected
identifyPch <- function(x, y = NULL, n = length(x), plot = FALSE, pch = 19, ...)
{
  xy <- xy.coords(x, y); x <- xy$x; y <- xy$y
  sel <- rep(FALSE, length(x))
  while(sum(sel) < n) {
    ans <- identify(x[!sel], y[!sel], labels = which(!sel), n = 1, plot = plot, ...)
    if(!length(ans)) break
    ans <- which(!sel)[ans]
    points(x[ans], y[ans], pch = pch)
    sel[ans] <- TRUE
  }
  ## return indices of selected points
  which(sel)
}

if(dev.interactive()) { ## use it
  x <- rnorm(50); y <- rnorm(50)
  plot(x,y); identifyPch(x,y) # how fast to get all?
}

```

image

```
z
useRaster
hcl.colorsn
```

```
image(x, ...)
```

```
## Default S3 method:
image(x, y, z, zlim, xlim, ylim,
      col = hcl.colors(12, "YlOrRd", rev = TRUE),
      add = FALSE, xaxs = "i", yaxs = "i", xlab, ylab,
      breaks, oldstyle = FALSE, useRaster, ...)
```

```
xy          zclistx$xx$yxyzz
z           NAxz
zlim        zz
xlimylim    xyxy
col         hcl.colorsgray.colors
add         TRUEimage
xaxsyaxs    "i"par
xlabylab    xy""
breaks
oldstyle    zlim[1]zlim[2]
useRaster   TRUE
...         plotaspaxesplot.window
```

```
xnrow(z)+1nrow(z)xyxyxy
add = TRUEpar("bg")
breakszlimcut
xlimylimxy
imagezf(x[i], y[j])
zuseRaster = TRUEpostscriptX11(type = "Xlib")windows()
```

```
useRastergetOption("preferRaster")dev.capabilities("rasterImage")$rasterImage
"yes""non-missing"
```


[filled.contour](#)[heatmap](#)[contour](#)[image](#)[levelplot](#)

[hcl.colors](#)[gray.colors](#)[shcl](#)[svpar](#)

[dev.capabilities](#)[useRaster](#) = TRUE

```
require("grDevices") # for colours
x <- y <- seq(-4*pi, 4*pi, length.out = 27)
r <- sqrt(outer(x^2, y^2, `+`))
image(z = z <- cos(r^2)*exp(-r/6), col = gray.colors(33))
image(z, axes = FALSE, main = "Math can be beautiful ...",
      xlab = expression(cos(r^2) * e^{-r/6}))
contour(z, add = TRUE, drawlabels = FALSE)

# Visualize as matrix. Need to transpose matrix and then flip it horizontally:
tf <- function(m) t(m)[, nrow(m):1]
imageM <- function(m, grid = max(dim(m)) <= 25, asp = (nrow(m)-1)/(ncol(m)-1), ...) {
  image(tf(m), asp=asp, axes = FALSE, ...)
  mAxis <- function(side, at, ...) # using 'j'
    axis(side, at=at, labels=as.character(j+1L), col="gray", col.axis=1, ...)
  n <- ncol(m); n1 <- n-1L; j <- 0L:n1; mAxis(1, at= j/n1)
  if(grid) abline(v = (0:n - .5)/n1, col="gray77", lty="dotted")
  n <- nrow(m); n1 <- n-1L; j <- 0L:n1; mAxis(2, at=1-j/n1, las=1)
  if(grid) abline(h = (0:n - .5)/n1, col="gray77", lty="dotted")
}
(m <- outer(1:5, 1:14))
imageM(m, main = "image(<5 x 14 matrix>) with rows and columns")
imageM(volcano)

# A prettier display of the volcano
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = hcl.colors(100, "terrain"), axes = FALSE)
contour(x, y, volcano, levels = seq(90, 200, by = 5),
      add = TRUE, col = "brown")
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)
```

layout

[layoutmat](#)

```
layout(mat, widths = rep.int(1, ncol(mat)),
      heights = rep.int(1, nrow(mat)), respect = FALSE)
```

```
layout.show(n = 1)
lcm(x)
```

```

mat           $N \times N \{1, \dots, N-1\}$ 
widths       lcm()
heights      widths
respect      mat[0]
n
x

```

```

iiimat
respect

```

```

layout.show(n)n
lcmwidthsheightslayout()

```

```

layout/N

```

```

par(mfrow)par(mfcol)split.screen

```

```

parmfrowmfcolmfgr

```

```

def.par <- par(no.readonly = TRUE) # save default, for resetting...

## divide the device into two rows and two columns
## allocate figure 1 all of row 1
## allocate figure 2 the intersection of column 2 and row 2
layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE))
## show the regions that have been allocated to each plot
layout.show(2)

## divide device into two rows and two columns
## allocate figure 1 and figure 2 as above
## respect relations between widths and heights
nf <- layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE), respect = TRUE)
layout.show(nf)

```

```

## create single figure which is 5cm square
nf <- layout(matrix(1), widths = lcm(5), heights = lcm(5))
layout.show(nf)

##-- Create a scatterplot with marginal histograms -----

x <- pmin(3, pmax(-3, stats::rnorm(50)))
y <- pmin(3, pmax(-3, stats::rnorm(50)))
xhist <- hist(x, breaks = seq(-3,3,0.5), plot = FALSE)
yhist <- hist(y, breaks = seq(-3,3,0.5), plot = FALSE)
top <- max(c(xhist$counts, yhist$counts))
xrange <- c(-3, 3)
yrange <- c(-3, 3)
nf <- layout(matrix(c(2,0,1,3),2,2,byrow = TRUE), c(3,1), c(1,3), TRUE)
layout.show(nf)

par(mar = c(3,3,1,1))
plot(x, y, xlim = xrange, ylim = yrange, xlab = "", ylab = "")
par(mar = c(0,3,1,1))
barplot(xhist$counts, axes = FALSE, ylim = c(0, top), space = 0)
par(mar = c(3,0,1,1))
barplot(yhist$counts, axes = FALSE, xlim = c(0, top), space = 0, horiz = TRUE)

par(def.par) #- reset to default

```

legend

[locator\(1\)](#)xy

```

legend(x, y = NULL, legend, fill = NULL, col = par("col"),
       border = "black", lty, lwd, pch,
       angle = 45, density = NULL, bty = "o", bg = par("bg"),
       box.lwd = par("lwd"), box.lty = par("lty"), box.col = par("fg"),
       pt.bg = NA, cex = 1, pt.cex = cex, pt.lwd = lwd,
       xjust = 0, yjust = 1, x.intersp = 1, y.intersp = 1,
       adj = c(0, 0.5), text.width = NULL, text.col = par("col"),
       text.font = NULL, merge = do.lines && has.pch, trace = FALSE,
       plot = TRUE, ncol = 1, horiz = FALSE, title = NULL,
       inset = 0, xpd, title.col = text.col[1], title.adj = 0.5,
       title.cex = cex[1], title.font = text.font[1],
       seg.len = 2)

```

xy	xy.coords
legend	\geq las.graphicsAnnot
fill	
col	
border	fill

```

ltylwd
pch          pointspoints
angle
density      NULLNA
bty          "o""n"
bg           bty != "n"
box.ltybox.lwdbox.col
             bty = "o"
pt.bg        pointsbg
cex          par("cex")pt.cex
pt.cex
pt.lwd       par("lwd")
xjust
yjust        xjust
x.intersp
y.intersp
adj          labels
text.width   "user"NULLstrwidth(legend)NAstrwidth(legend)
text.col
text.font    text
merge        TRUETRUE
trace        TRUElegend
plot         FALSE
ncol
horiz        TRUEhorizncol
title        as.graphicsAnnot
inset
xpd          xpd
title.col    titletext.col[1]
title.adj    titlepar("adj")
title.cex    cex[1]
title.font   text.font[1]text
seg.len      ltylwd

xylegendlegendylegend
xy.coords
x"bottomright""bottom""bottomleft""left""topleft""top""topright""right""center"
insetxy
colpchltymergelty0lwdNApchNA
pt.bg

-31:-1NA""

```

```

rect
      wh
      lefttop
text
      x, y length(legend)

```

`plotbarplotlegend()``text`

```

## Run the example in '?matplot' or the following:
leg.txt <- c("Setosa      Petals", "Setosa      Sepals",
            "Versicolor Petals", "Versicolor Sepals")
y.leg <- c(4.5, 3, 2.1, 1.4, .7)
cexv <- c(1.2, 1, 4/5, 2/3, 1/2)
matplot(c(1, 8), c(0, 4.5), type = "n", xlab = "Length", ylab = "Width",
        main = "Petal and Sepal Dimensions in Iris Blossoms")
for (i in seq(cexv)) {
  text (1, y.leg[i] - 0.1, paste("cex=", formatC(cexv[i])), cex = 0.8, adj = 0)
  legend(3, y.leg[i], leg.txt, pch = "sSvV", col = c(1, 3), cex = cexv[i])
}
## cex *vector* [in R <= 3.5.1 has 'if(xc < 0)' w/ length(xc) == 2]
legend("right", leg.txt, pch = "sSvV", col = c(1, 3),
      cex = 1+(-1:2)/8, trace = TRUE)# trace: show computed lengths & coords

## 'merge = TRUE' for merging lines & points:
x <- seq(-pi, pi, length.out = 65)
for(reverse in c(FALSE, TRUE)) { ## normal *and* reverse axes:
  F <- if(reverse) rev else identity
  plot(x, sin(x), type = "l", col = 3, lty = 2,
       xlim = F(range(x)), ylim = F(c(-1.2, 1.8)))
  points(x, cos(x), pch = 3, col = 4)
  lines(x, tan(x), type = "b", lty = 1, pch = 4, col = 6)
  title("legend('top', lty = c(2, -1, 1), pch = c(NA, 3, 4), merge = TRUE)",
        cex.main = 1.1)
  legend("top", c("sin", "cos", "tan"), col = c(3, 4, 6),
        text.col = "green4", lty = c(2, -1, 1), pch = c(NA, 3, 4),
        merge = TRUE, bg = "gray90", trace=TRUE)

} # for(..)

## right-justifying a set of labels: thanks to Uwe Ligges
x <- 1:5; y1 <- 1/x; y2 <- 2/x
plot(rep(x, 2), c(y1, y2), type = "n", xlab = "x", ylab = "y")
lines(x, y1); lines(x, y2, lty = 2)

```

```

temp <- legend("topright", legend = c(" ", " "),
              text.width = strwidth("1,000,000"),
              lty = 1:2, xjust = 1, yjust = 1, inset = 1/10,
              title = "Line Types", title.cex = 0.5, trace=TRUE)
text(temp$rect$left + temp$rect$w, temp$text$y,
      c("1,000", "1,000,000"), pos = 2)

##--- log scaled Examples -----
leg.txt <- c("a one", "a two")

par(mfrow = c(2, 2))
for(ll in c("", "x", "y", "xy")) {
  plot(2:10, log = ll, main = paste0("log = '", ll, "'"))
  abline(1, 1)
  lines(2:3, 3:4, col = 2)
  points(2, 2, col = 3)
  rect(2, 3, 3, 2, col = 4)
  text(c(3,3), 2:3, c("rect(2,3,3,2, col=4)",
                      "text(c(3,3),2:3,\"c(rect(...)\")\"", adj = c(0, 0.3))
  legend(list(x = 2,y = 8), legend = leg.txt, col = 2:3, pch = 1:2,
            lty = 1) #, trace = TRUE)
} # ^^^^^^ to force lines -> automatic merge=TRUE
par(mfrow = c(1,1))

##-- Math expressions: -----
x <- seq(-pi, pi, length.out = 65)
plot(x, sin(x), type = "l", col = 2, xlab = expression(phi),
      ylab = expression(f(phi)))
abline(h = -1:1, v = pi/2*(-6:6), col = "gray90")
lines(x, cos(x), col = 3, lty = 2)
ex.cs1 <- expression(plain(sin) * phi, paste("cos", phi)) # 2 ways
utils::str(legend(-3, .9, ex.cs1, lty = 1:2, plot = FALSE,
                  adj = c(0, 0.6))) # adj y !
legend(-3, 0.9, ex.cs1, lty = 1:2, col = 2:3, adj = c(0, 0.6))

require(stats)
x <- rexp(100, rate = .5)
hist(x, main = "Mean and Median of a Skewed Distribution")
abline(v = mean(x), col = 2, lty = 2, lwd = 2)
abline(v = median(x), col = 3, lty = 3, lwd = 2)
ex12 <- expression(bar(x) == sum(over(x[i], n), i == 1, n),
                   hat(x) == median(x[i], i == 1, n))
utils::str(legend(4.1, 30, ex12, col = 2:3, lty = 2:3, lwd = 2))

## 'Filled' boxes -- see also example(barplot) which may call legend(*, fill=)
barplot(VADeaths)
legend("topright", rownames(VADeaths), fill = gray.colors(nrow(VADeaths)))

## Using 'ncol'
x <- 0:64/64
for(R in c(identity, rev)) { # normal *and* reverse x-axis works fine:
  x1 <- R(range(x)); x1 <- x1[1]
  matplot(x, outer(x, 1:7, function(x, k) sin(k * pi * x)), xlim=x1,
          type = "o", col = 1:7, ylim = c(-1, 1.5), pch = "*")
  op <- par(bg = "antiquewhite1")
  legend(x1, 1.5, paste("sin(", 1:7, "pi * x)"), col = 1:7, lty = 1:7,

```

```

        pch = "*", ncol = 4, cex = 0.8)
legend("bottomright", paste("sin(", 1:7, "pi * x)"), col = 1:7, lty = 1:7,
      pch = "*", cex = 0.8)
legend(x1, -.1, paste("sin(", 1:4, "pi * x)"), col = 1:4, lty = 1:4,
      ncol = 2, cex = 0.8)
legend(x1, -.4, paste("sin(", 5:7, "pi * x)"), col = 4:6, pch = 24,
      ncol = 2, cex = 1.5, lwd = 2, pt.bg = "pink", pt.cex = 1:3)
par(op)

} # for(..)

## point covering line :
y <- sin(3*pi*x)
plot(x, y, type = "l", col = "blue",
     main = "points with bg & legend(*, pt.bg)")
points(x, y, pch = 21, bg = "white")
legend(.4,1, "sin(c x)", pch = 21, pt.bg = "white", lty = 1, col = "blue")

## legends with titles at different locations
plot(x, y, type = "n")
legend("bottomright", "(x,y)", pch=1, title= "bottomright")
legend("bottom",      "(x,y)", pch=1, title= "bottom")
legend("bottomleft",  "(x,y)", pch=1, title= "bottomleft")
legend("left",        "(x,y)", pch=1, title= "left")
legend("topleft",     "(x,y)", pch=1, title= "topleft", inset = .05, inset = .05)
legend("top",         "(x,y)", pch=1, title= "top")
legend("topright",    "(x,y)", pch=1, title= "topright", inset = .02, inset = .02)
legend("right",       "(x,y)", pch=1, title= "right")
legend("center",      "(x,y)", pch=1, title= "center")

# using text.font (and text.col):
op <- par(mfrow = c(2, 2), mar = rep(2.1, 4))
c6 <- terrain.colors(10)[1:6]
for(i in 1:4) {
  plot(1, type = "n", axes = FALSE, ann = FALSE); title(paste("text.font =",i))
  legend("top", legend = LETTERS[1:6], col = c6,
        ncol = 2, cex = 2, lwd = 3, text.font = i, text.col = c6)
}
par(op)

# using text.width for several columns
plot(1, type="n")
legend("topleft", c("This legend", "has", "equally sized", "columns."),
      pch = 1:4, ncol = 4)
legend("bottomleft", c("This legend", "has", "optimally sized", "columns."),
      pch = 1:4, ncol = 4, text.width = NA)
legend("right", letters[1:4], pch = 1:4, ncol = 4,
      text.width = 1:4 / 50)

```

lines

```

lines(x, ...)

## Default S3 method:
lines(x, y = NULL, type = "l", ...)

xy
type          typeplot.default
...           parltylwdcoltype = "b"pchpointslendljoinlmitre

```

```

xyxy.coords
NANaxy
type = "h"col
lwd

```

```

lines.formulapointstype %in% c("p","b","o")plotplot.xy
abline
parlty

```

```

# draw a smooth line through a scatter plot
plot(cars, main = "Stopping Distance versus Speed")
lines(stats::lowess(cars))

```

locator

```

locator(n = 512, type = "n", ...)

n
type          "n""p""l""o""p""o""l""o"
...           type != "n"

```



```
locatorX11windowsquartz
n
X11quartzESC
plot.defaultttype
locatoroptions(locatorBell = FALSE)
locatorlocator

xypar("usr")
```

```
identifygrid.locator
dev.capabilities
```

matplot

```
matplot(x, y, type = "p", lty = 1:5, lwd = 1, lend = par("lend"),
        pch = NULL,
        col = 1:6, cex = NULL, bg = NA,
        xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
        log = "", ..., add = FALSE, verbose = getOption("verbose"))
```

```
matpoints(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL,
          col = 1:6, ...)
```

```
matlines (x, y, type = "l", lty = 1:5, lwd = 1, pch = NULL,
          col = 1:6, ...)
```

xy	yx1:nNAclass(.)xy"Date"
type	yplottypetypetype"pl"
ltylwdlend	
pch	points
col	
cex	par("cex")NULL1.0
bg	pch = 21:25pointsNAplot.xy
xlabylab	plot
xlimylim	plot
log...	parplotplot.defaultpanel.firstparttitle
add	TRUEpointslines
verbose	TRUE

```
matplot(x,y, ...)

plot(x[,1], y[,1], ...)add = TRUE
lines(x[,j], y[,j], ...)

class(.)xyplot()lines()
```

```
xyxyxy
col, cex, lty, lwd
lty = 1
```

```
matplotmatpointsmatlines
```

```
plotpointslinesmatrixpar
```

```
require(grDevices)
matplot((-4:5)^2, main = "Quadratic") # almost identical to plot(*)
sines <- outer(1:20, 1:4, function(x, y) sin(x / 20 * pi * y))
matplot(sines, pch = 1:4, type = "o", col = rainbow(ncol(sines)))
matplot(sines, type = "b", pch = 21:23, col = 2:5, bg = 2:5,
        main = "matplot(..., pch = 21:23, bg = 2:5)")

x <- 0:50/50
matplot(x, outer(x, 1:8, function(x, k) sin(k*pi * x)),
        ylim = c(-2,2), type = "plobcsSh",
        main= "matplot(,type = \"plobcsSh\" )")
## pch & type = vector of 1-chars :
matplot(x, outer(x, 1:4, function(x, k) sin(k*pi * x)),
        pch = letters[1:4], type = c("b","p","o"))

lends <- c("round","butt","square")
matplot(matrix(1:12, 4), type="c", lty=1, lwd=10, lend=lends)
text(cbind(2.5, 2*c(1,3,5)-.4), lends, col= 1:3, cex = 1.5)

table(iris$Species) # is data.frame with 'Species' factor
iS <- iris$Species == "setosa"
iV <- iris$Species == "versicolor"
op <- par(bg = "bisque")
matplot(c(1, 8), c(0, 4.5), type = "n", xlab = "Length", ylab = "Width",
        main = "Petal and Sepal Dimensions in Iris Blossoms")
matpoints(iris[iS,c(1,3)], iris[iS,c(2,4)], pch = "sS", col = c(2,4))
matpoints(iris[iV,c(1,3)], iris[iV,c(2,4)], pch = "vV", col = c(2,4))
legend(1, 4, c("Setosa Petals", "Setosa Sepals",
               "Versicolor Petals", "Versicolor Sepals"),
      pch = "sSvV", col = rep(c(2,4), 2))
```

```

matplot(iris3[, "Petal L.",], iris3[, "Petal W.",], pch = "SCV",
        col = rainbow(3, start = 0.8, end = 0.1),
        sub = paste(c("S", "C", "V"), dimnames(iris3)[[3]]),
              sep = "=", collapse= ", "),
        main = "Fisher's Iris Data")
par(op)

## 'x' a "Date" vector :
nd <- length(dv <- seq(as.Date("1959-02-21"), by = "weeks", length.out = 100))
mSC <- cbind(I=1, sin=sin(pi*(1:nd)/8), cos=cos(pi*(1:nd)/8))
matplot(dv, mSC, type = "b", main = "matplot(<Date>, y)")

## 'x' a "POSIXct" date-time vector :
ct <- seq(c(ISOdate(2000,3,20)), by = "15 mins", length.out = 100)
matplot(ct, mSC, type = "b", main = "matplot(<POSIXct>, y)")
## or the same with even more axis flexibility:
matplot(ct, mSC, type = "b", main = "matplot(<POSIXct>, y)", xaxt="n")
Axis(ct, side=1, at = ct[1+4*(0:24)])

iS <- 1:3 # indices of subset
matplot(gait[, iS, 1], gait[, iS, 2], pch = "123", type = "b",
        col = rainbow(3, start = 0.8, end = 0.1),
        sub = paste(iS, dimnames(gait)[[2]][iS],
                    sep = "=", collapse= ", "),
        xlab = "Hip angle", ylab = "Knee angle",
        main = "Hip and knee angle while walking")

## Also works for data frame columns:
matplot(iris[1:50,1:4])

```

mosaicplot

```

mosaicplot(x, ...)

## Default S3 method:
mosaicplot(x, main = deparse1(substitute(x)),
          sub = NULL, xlab = NULL, ylab = NULL,
          sort = NULL, off = NULL, dir = NULL,
          color = NULL, shade = FALSE, margin = NULL,
          cex.axis = 0.66, las = par("las"), border = NULL,
          type = c("pearson", "deviance", "FT"), ...)

## S3 method for class 'formula'
mosaicplot(formula, data = NULL, ...,
          main = deparse1(substitute(data)), subset,
          na.action = stats::na.omit)

```

```

x          dimnames(x)table()
main
sub
xlabylab   names(dimnames(X))X
sort       1:length(dim(x))
off
dir        "v""h"
color      shadeFALSENULLcolor = TRUEgrey.colorscolor = FALSE
shade      shadeFALSEshade = TRUE
margin     loglin
cex.axis   par("cex")
las        par
border     polygon
type       "pearson" $\chi^2$ "deviance" $\chi^2$ "FT"
formula    y ~ x
data       formula
...
subset
na.action  NA NA na.action

```

```

mosaicplot.defaultmosaicplot.formula

```

```

data"table""f"table"formula
datasubsetformula

```

```

na.actiondata
mosaic

```

```

<john.emerson@yale.edu>

```

```

assocplotloglin

```

```

require(stats)
mosaicplot(Titanic, main = "Survival on the Titanic", color = TRUE)
## Formula interface for tabulated data:
mosaicplot(~ Sex + Age + Survived, data = Titanic, color = TRUE)

mosaicplot(HairEyeColor, shade = TRUE)
## Independence model of hair and eye color and sex. Indicates that
## there are more blue eyed blonde females than expected in the case
## of independence and too few brown eyed blonde females.
## The corresponding model is:
fm <- loglin(HairEyeColor, list(1, 2, 3))
pchisq(fm$pearson, fm$df, lower.tail = FALSE)

mosaicplot(HairEyeColor, shade = TRUE, margin = list(1:2, 3))
## Model of joint independence of sex from hair and eye color. Males
## are underrepresented among people with brown hair and eyes, and are
## overrepresented among people with brown hair and blue eyes.
## The corresponding model is:
fm <- loglin(HairEyeColor, list(1:2, 3))
pchisq(fm$pearson, fm$df, lower.tail = FALSE)

## Formula interface for raw data: visualize cross-tabulation of numbers
## of gears and carburettors in Motor Trend car data.
mosaicplot(~ gear + carb, data = mtcars, color = TRUE, las = 1)
# color recycling
mosaicplot(~ gear + carb, data = mtcars, color = 2:3, las = 1)

```

mtext

```

mtext(text, side = 3, line = 0, outer = FALSE, at = NA,
      adj = NA, padj = NA, cex = NA, col = NA, font = NA, ...)

```

text	as.graphicsAnnot
side	
line	
outer	
at	atadj
adj	adj = 0adj = 1 adjpar("las")
padj	adjpadj = 0padj = 1 padjpar("las")
cex	NULLNA1.0par("cex")par("mfrow")par("mfcol")
col	NAPar("col")
font	NAPar("font")
...	par familylasxpouter = TRUE

```
adjtextadj = 0.5outer = TRUE
lassrtatlassrt
adjat
"ylbias"par
```

```
titletextplotparplotmath
```

```
plot(1:10, (-4:5)^2, main = "Parabola Points", xlab = "xlab")
mtext("10 of them")
for(s in 1:4)
  mtext(paste("mtext(..., line= -1, {side, col, font} = ", s,
    ", cex = ", (1+s)/2, ")"), line = -1,
    side = s, col = s, font = s, cex = (1+s)/2)
mtext("mtext(..., line= -2)", line = -2)
mtext("mtext(..., line= -2, adj = 0)", line = -2, adj = 0)
##--- log axis :
plot(1:10, exp(1:10), log = "y", main = "log = \"y\"", xlab = "xlab")
for(s in 1:4) mtext(paste("mtext(...,side=", s ,")"), side = s)

##--- illustrating padj behavior :
plot(0, axes=FALSE, ann=FALSE, frame.plot=TRUE)
for(si in 1:4) mtext(c("padj=0", "-----", "padj=1"),
  side = si, padj = c(0, 0.5, 1))
```

pairs

```
pairs(x, ...)

## S3 method for class 'formula'
pairs(formula, data = NULL, ..., subset,
  na.action = stats::na.pass)

## Default S3 method:
pairs(x, labels, panel = points, ...,
```

```

horInd = 1:nc, verInd = 1:nc,
lower.panel = panel, upper.panel = panel,
diag.panel = NULL, text.panel = textPanel,
label.pos = 0.5 + has.diag/3, line.main = 3,
cex.labels = NULL, font.labels = 1,
row1attop = TRUE, gap = 1, log = "",
horOdd = !row1attop, verOdd = !row1attop)

```

```

x                data.matrix
formula          ~ x + y + z
data             formula
subset
na.action        NAna.action = na.omit
labels
panel            function(x, y, ...)
...
                plotmainpar("oma")
horIndverInd
lower.panelupper.panel
                NULL
diag.panel       function(x, ...)
text.panel       function(x, y, labels, cex, font, ...)
label.pos        y
line.main        mainline.mainlinemtext()omaline.main
cex.labelsfont.labels

row1attop
gap
log              plot.defaultlog = "xy"
horOddverOdd     logical

```

```

ijx[,i]x[,j]xxytext.panel(x, y)NULL
pchcol
omapairs.default
plotboxplot
na.action = na.omit
horIndverInd

```

```

pairs(iris[1:4], main = "Anderson's Iris Data -- 3 species",
      pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])

## formula method, "graph" layout (row 1 at bottom):
pairs(~ Fertility + Education + Catholic, data = swiss, rowlattice=FALSE,
      subset = Education < 20, main = "Swiss data, Education < 20")

pairs(USJudgeRatings, gap=1/10) # (gap: not wasting plotting area)
## show only lower triangle (and suppress labeling for whatever reason):
pairs(USJudgeRatings, text.panel = NULL, upper.panel = NULL)

## put histograms on the diagonal
panel.hist <- function(x, ...)
{
  usr <- par("usr")
  par(usr = c(usr[1:2], 0, 1.5) )
  h <- hist(x, plot = FALSE)
  breaks <- h$breaks; nB <- length(breaks)
  y <- h$counts; y <- y/max(y)
  rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}
pairs(USJudgeRatings[1:5], panel = panel.smooth,
      cex = 1.5, pch = 24, bg = "light blue", horOdd=TRUE,
      diag.panel = panel.hist, cex.labels = 2, font.labels = 2)

## put (absolute) correlations on the upper panels,
## with size proportional to the correlations.
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(USJudgeRatings, lower.panel = panel.smooth, upper.panel = panel.cor,
      gap=0, rowlattice=FALSE)

pairs(iris[-5], log = "xy") # plot all variables on log scale
pairs(iris, log = 1:4, # log the first four
      main = "Lengths and Widths in [log]", line.main=1.5, oma=c(2,2,3,2))

```

panel.smooth

panelcoplotpairs

```

panel.smooth(x, y, col = par("col"), bg = NA, pch = par("pch"),
             cex = 1, col.smooth = 2, span = 2/3, iter = 3,
             ...)

```



```

xy
colbgpchcex    pointspar
col.smooth     lines
span           flowess
iter           lowess
...            lines

```

`coplotpairs``panel.smooth``lowess`

```

pairs(swiss, panel = panel.smooth, pch = ".") # emphasize the smooths
pairs(swiss, panel = panel.smooth, lwd = 2, cex = 1.5, col = 4) # hmm...

```

`par`

```

parpartag = value

```

```

par(..., no.readonly = FALSE)

```

```

<highlevel plot> (...., <tag> = <value>)

```

```

...            tag = value
no.readonly    TRUEpar()

```

```

paroptions("device")

```

```

par

```

```

par()par(no.readonly = TRUE)graphics:::.Pars

```

```

"cin""cra""csi""cxy""din""page"

```

```

par()

```

```

"ask"

```

```

"fig""fin"

```

```

"lheight"

```

```

"mai""mar""mex""mfcol""mfrow""mfg"

```

```

"new"

```

```

"oma""omd""omi"

```

```

"pin""plt""ps""pty"

```

```

"usr"

```

```

"xlog" "ylog"
"ylbias"

...plot.defaultplot.windowpointslinesablineaxis titletextmttextsegmentssymbols
arrowspolygonrectboxcontourfilled.contourimagebgcexcolltylwdpch
pointsizecracincxycsimaromalheightttextstrheight
plot

parpar(no.readonly = TRUE)

adj adjtextmttexttitle00.51[0,1]
  adjtextadj = c(x, y)textmttexttitle
ann FALSEplot.default
ask TRUE
  devAskNewPage
bg par()new = FALSEbg"white"
  plot.defaultpoints
bty boxbty"o""l""7""c""u""j""n"
cex 1mfrow
  plot.defaultpointstext
cex.axis cex
cex.lab cex
cex.main cex
cex.sub cex
cin (width, height)cra
col
  linestext
col.axis "black"
col.lab "black"
col.main "black"
col.sub "black"
cra (width, height)cin
crt srt
csi par("cin")[2]
cxy (width, height)par("cxy")par("cin")/par("pin")c(strwidth(ch),
  strheight(ch))ch
din (width, height)dev.size
err
family "" "serif" "sans" "mono" "Hershey" text

```

```

fg par()col"black"

fig c(x1, x2, y1, y2)new = TRUE

fin (width, height)

font family

font.axis

font.lab

font.main

font.sub

lab c(x, y, len)xylenc(5, 5, 7)len

las

```

```

mtextsrtpar

lend

0 "round"
1 "butt"
2 "square"

lheight textstrheight

ljoin

0 "round"
1 "mitre"
2 "bevel"

lmitre

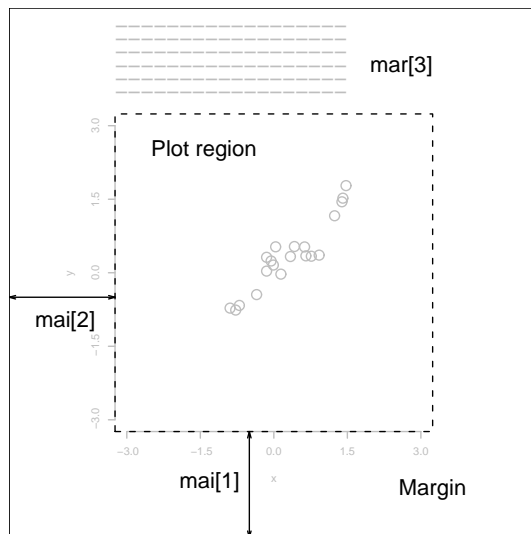
lty "blank""solid""dashed""dotted""dotdash""longdash""twodash""blank"
c(1:9, "A":"F")
linessegments

lwd 1

linessegmentsNANaN0

mai c(bottom, left, top, right)

```



```
mar c(bottom, left, top, right)c(5, 4, 4, 2) + 0.1
```

```
mex mexcsimarmaiomaomi
```

```
1cex
```

```
mfgcol, mfrow c(nr, nc)nrncmfcolmfrow
```

```
"cex"
```

```
cexmex1
```

```
layoutsplitted.screen
```

```
mfg c(i, j)ijmfcolmfrow
```

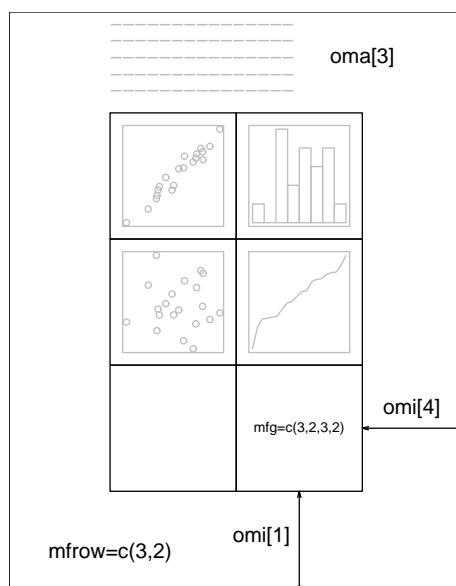
```
c(i, j, nr, nc)nrnc
```

```
mfg mexmfg[1]titlemfg[2:3]axisc(3, 1, 0)
```

```
mkg pch
```

```
new FALSETRUEplot.newnew = TRUE
```

```
oma c(bottom, left, top, right)
```



```

omd c(x1, x2, y1, y2)[0,1]
omi c(bottom, left, top, right)
page plot.newFALSE
pch pointsNANULL
    points
pin (width, height)
plt c(x1, x2, y1, y2)
ps pointsizearmaiomaomi

pty "s""m"
smo
srt crttext
tck tck >= 0.5tck = 1tck = NA tcl = -0.5
tcl -0.5tcl = NA tck = -0.01
usr c(x1, x2, y1, y2)par("xlog")10 ^ par("usr")[1:2]
xaxp c(x1, x2, n)par("xlog")nn1:3x1x210 ^ par("usr")[1:2]"usr"
    10j
    k10j k ∈ {1, 5}
    k10j k ∈ {1, 2, 5}
    axTicks()
    plot.windowpar("usr")npar("lab")axis

xaxs "r""i""e""s""d"xlim
    "r"
    "i"
    "s"
    "e""s"
    "d"
    "r""i"
xaxt "n""s""l""t""s""n"
xlog logplot.defaultTRUEplot(*, log = "x")FALSE
xpd NAFALSETRUENAclip
yaxp c(y1, y2, n)xaxp
yaxs xaxs
yaxt "n"
ylbias axismtext0.20.2x11windows()
ylog xlog

"red"colors"#RRGGBB"#RRGGBB00FF"#rrggbb""#rrggbbaa""#rgb""#rgba""#rgb""#rrggbb"
palette10200

"transparent"NA"transparent"textmtext

rgbhsvhclgrayrainbow

```

```
lty"33""3313"lwdlwd = 1
lty = 2:6c("44", "13", "1343", "73", "2262")
NA lty
```

```
mairpinpltpty
```

[plot.defaultcolscloptionsx11pdfpostscriptlayoutsplitted.screen](#)

```
op <- par(mfrow = c(2, 2), # 2 x 2 pictures on one plot
          pty = "s")      # square plotting region,
                          # independent of device size

## At end of plotting, reset to previous settings:
par(op)

## Alternatively,
op <- par(no.readonly = TRUE) # the whole list of settable par's.
## do lots of plotting and par(.) calls, then reset:
par(op)
## Note this is not in general good practice

par("ylog") # FALSE
plot(1 : 12, log = "y")
par("ylog") # TRUE

plot(1:2, xaxs = "i") # 'inner axis' w/o extra space
par(c("usr", "xaxp"))

( nr.prof <-
c(prof.pilots = 16, lawyers = 11, farmers = 10, salesmen = 9, physicians = 9,
  mechanics = 6, policemen = 6, managers = 6, engineers = 5, teachers = 4,
  housewives = 3, students = 3, armed.forces = 1))
par(las = 3)
barplot(rbind(nr.prof)) # R 0.63.2: shows alignment problem
par(las = 0) # reset to default

require(grDevices) # for gray
## 'fg' use:
plot(1:12, type = "b", main = "'fg' : axes, ticks and box in gray",
     fg = gray(0.7), bty = "n", sub = R.version.string)

ex <- function() {
  old.par <- par(no.readonly = TRUE) # all par settings which
                                     # could be changed.
  on.exit(par(old.par))
```

```

## ...
## ... do lots of par() settings and plots
## ...
invisible() #-- now, par(old.par) will be executed
}
ex()

## Line types
showLty <- function(ltys, xoff = 0, ...) {
  stopifnot((n <- length(ltys)) >= 1)
  op <- par(mar = rep(.5,4)); on.exit(par(op))
  plot(0:1, 0:1, type = "n", axes = FALSE, ann = FALSE)
  y <- (n:1)/(n+1)
  clty <- as.character(ltys)
  mytext <- function(x, y, txt)
    text(x, y, txt, adj = c(0, -.3), cex = 0.8, ...)
  abline(h = y, lty = ltys, ...); mytext(xoff, y, clty)
  y <- y - 1/(3*(n+1))
  abline(h = y, lty = ltys, lwd = 2, ...)
  mytext(1/8+xoff, y, paste(clty, " lwd = 2"))
}
showLty(c("solid", "dashed", "dotted", "dotdash", "longdash", "twodash"))
par(new = TRUE) # the same:
showLty(c("solid", "44", "13", "1343", "73", "2262"), xoff = .2, col = 2)
showLty(c("11", "22", "33", "44", "12", "13", "14", "21", "31"))

```

persp

persp

persp(x, ...)

Default S3 method:

```

persp(x = seq(0, 1, length.out = nrow(z)),
      y = seq(0, 1, length.out = ncol(z)),
      z, xlim = range(x), ylim = range(y),
      zlim = range(z, na.rm = TRUE),
      xlab = NULL, ylab = NULL, zlab = NULL,
      main = NULL, sub = NULL,
      theta = 0, phi = 15, r = sqrt(3), d = 1,
      scale = TRUE, expand = 1,
      col = "white", border = NULL, ltheta = -135, lphi = 0,
      shade = NA, box = TRUE, axes = TRUE, nticks = 5,
      ticktype = "simple", ...)

```

xy zxlistx\$xx\$yxy

z NAxz

xlimylimzlim

xlabylabzlab

```

mainsub      title
thetaphi     thetaphi
r
d            d
scale        scaleTRUEscaleFALSE
expand       z0 < expand < 1z
col          (nx - 1)(ny - 1)
border       NULLpar("fg")NA
lthetalphi   lthetalphilthetalphi
shade        ((1+d)/2)^shadedshade
box          TRUE
axes         TRUEboxFALSE
ticktype     "simple""detailed"
nticks       ticktype"simple"
...          par

thetaphithetaphithetaphi
"persp"setHook
perspzf(x[i], y[j])
ticktype = "detailed""cex.lab""font.lab""cex.axis""font.axis"

```

```

persp()VT4 × 4(x, y, z)(x, y, z, t)lines()points()trans3d()

```

```

contourimagetrans3d

```

```

require(grDevices) # for trans3d
## More examples in demo(persp) !!
## -----

# (1) The Obligatory Mathematical surface.
#   Rotated sinc function.

x <- seq(-10, 10, length.out = 30)
y <- x
f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z <- outer(x, y, f)
op <- par(bg = "white")

```



```

persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
      ltheta = 120, shade = 0.75, ticktype = "detailed",
      xlab = "X", ylab = "Y", zlab = "Sinc( r )", cex.axis = 0.8
) -> res
round(res, 3)

# (2) Add to existing persp plot - using trans3d() :

xE <- c(-10,10); xy <- expand.grid(xE, xE)
points(trans3d(xy[,1], xy[,2], z = 6,          pmat = res), col = 2, pch = 16)
lines (trans3d(x,      y = 10, z = 6 + sin(x), pmat = res), col = 3)

phi <- seq(0, 2*pi, length.out = 201)
r1 <- 7.725 # radius of 2nd maximum
xr <- r1 * cos(phi)
yr <- r1 * sin(phi)
lines(trans3d(xr,yr, f(xr,yr), res), col = "pink", lwd = 2)
## (no hidden lines)

# (3) Visualizing a simple DEM model

z <- 2 * volcano          # Exaggerate the relief
x <- 10 * (1:nrow(z))     # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))     # 10 meter spacing (E to W)
## Don't draw the grid lines : border = NA
par(bg = "slategray")
persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75, border = NA, box = FALSE)

# (4) Surface colours corresponding to z-values

par(bg = "white")
x <- seq(-1.95, 1.95, length.out = 30)
y <- seq(-1.95, 1.95, length.out = 35)
z <- outer(x, y, function(a, b) a*b^2)
nrz <- nrow(z)
ncz <- ncol(z)
# Create a function interpolating colors in the range of specified colors
jet.colors <- colorRampPalette( c("blue", "green") )
# Generate the desired number of colors from this palette
nbcol <- 100
color <- jet.colors(nbcol)
# Compute the z-value at the facet centres
zfacet <- z[-1, -1] + z[-1, -ncz] + z[-nrz, -1] + z[-nrz, -ncz]
# Recode facet z-values into color indices
facetcol <- cut(zfacet, nbcol)
persp(x, y, z, col = color[facetcol], phi = 30, theta = -30)

par(op)

```

```
pie(x, labels = names(x), edges = 200, radius = 0.8,
    clockwise = FALSE, init.angle = if(clockwise) 90 else 0,
    density = NULL, angle = 45, col = NULL, border = NULL,
    lty = NULL, main = NULL, ...)
```

x	x
labels	<code>as.graphicsAnnotNA</code>
edges	
radius	-11
clockwise	
init.angle	clockwiseinit.angle
density	NULLdensity
angle	
col	densitypar("fg")
borderlty	<code>polygon</code>
main	
...	pie

`dotchart`

```
require(grDevices)
pie(rep(1, 24), col = rainbow(24), radius = 0.9)

pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
names(pie.sales) <- c("Blueberry", "Cherry",
  "Apple", "Boston Cream", "Other", "Vanilla Cream")
pie(pie.sales) # default colours
pie(pie.sales, col = c("purple", "violetred1", "green3",
  "cornsilk", "cyan", "white"))
pie(pie.sales, col = gray(seq(0.4, 1.0, length.out = 6)))
pie(pie.sales, density = 10, angle = 15 + 10 * 1:6)
pie(pie.sales, clockwise = TRUE, main = "pie(*, clockwise = TRUE)")
segments(0, 0, 0, 1, col = "red", lwd = 2)
text(0, 1, "init.angle = 90", col = "red")

n <- 200
```

```
pie(rep(1, n), labels = "", col = rainbow(n), border = NA,  
    main = "pie(*, labels=\"\\\", col=rainbow(n), border=NA,..")  
  
## Another case showing pie() is rather fun than science:  
## (original by FinalBackwardsGlance on http://imgur.com/gallery/wWrpu4X)  
pie(c(Sky = 78, "Sunny side of pyramid" = 17, "Shady side of pyramid" = 5),  
    init.angle = 315, col = c("deepskyblue", "yellow", "yellow3"), border = FALSE)
```

`plot.data.frame`

`plot.data.frame`[plot](#)

```
## S3 method for class 'data.frame'  
plot(x, ...)
```

```
x                data.frame  
...              stripchartplot.defaultpairs
```

[data.matrixpairs](#)

[stripchart](#)

[data.frame](#)

```
plot(OrchardSprays[1], method = "jitter")  
plot(OrchardSprays[c(4,1)])  
plot(OrchardSprays)
```

```
plot(iris)  
plot(iris[5:4])  
plot(women)
```

plot.default

```
## Default S3 method:
plot(x, y = NULL, type = "p", xlim = NULL, ylim = NULL,
      log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
      ann = par("ann"), axes = TRUE, frame.plot = axes,
      panel.first = NULL, panel.last = NULL, asp = NA,
      xgap.axis = NA, ygap.axis = NA,
      ...)
```

xy	xyxy.coords
type	plot"p""l""b""c""o""s""S""h""n"
xlim	x1 > x2 NULL
ylim	
log	"x""y""xy""yx"
main	title
sub	
xlab	x
ylab	y
ann	
axes	"xaxt""yaxt"
frame.plot	
panel.first	plot
panel.last	panel.first
asp	y/xplot.window
xgap.axis	
ygap.axis	x/ygap.axisaxis()axes
...	par

```
col
bg pointspar("bg")
pch points
cex par("cex")NULLNA1.0
lty par
cex.maincol.labfont.sub titlepar
lwd par
```

```

panel.firstpanel.lastpairsplot.defaultplot.newplot.windowplot.xyaxisboxtitle
plot(type = "n")
plot

```

```

plotplot.windowxy.coordssmoothScatter

```

```

Speed <- cars$speed
Distance <- cars$dist
plot(Speed, Distance, panel.first = grid(8, 8),
     pch = 0, cex = 1.2, col = "blue")
plot(Speed, Distance,
     panel.first = lines(stats::lowess(Speed, Distance), lty = "dashed"),
     pch = 0, cex = 1.2, col = "blue")

## Show the different plot types
x <- 0:12
y <- sin(pi/5 * x)
op <- par(mfrow = c(3,3), mar = .1+ c(2,2,3,1))
for (tp in c("p","l","b", "c","o","h", "s","S","n")) {
  plot(y ~ x, type = tp, main = paste0("plot(*, type = \"", tp, "\")"))
  if(tp == "S") {
    lines(x, y, type = "s", col = "red", lty = 2)
    mtext("lines(*, type = \"s\", ...)", col = "red", cex = 0.8)
  }
}
par(op)

##--- Log-Log Plot with custom axes
lx <- seq(1, 5, length.out = 41)
yl <- expression(e^{ -frac(1,2) * {log[10](x)}^2})
y <- exp(-.5*lx^2)
op <- par(mfrow = c(2,1), mar = par("mar")-c(1,0,2,0), mgp = c(2, .7, 0))
plot(10^lx, y, log = "xy", type = "l", col = "purple",
     main = "Log-Log plot", ylab = yl, xlab = "x")
plot(10^lx, y, log = "xy", type = "o", pch = ".", col = "forestgreen",
     main = "Log-Log plot with custom axes", ylab = yl, xlab = "x",
     axes = FALSE, frame.plot = TRUE)
my.at <- 10^(1:5)
axis(1, at = my.at, labels = formatC(my.at, format = "fg"))
e.y <- -5:-1 ; at.y <- 10^e.y
axis(2, at = at.y, col.axis = "red", las = 1,
     labels = as.expression(lapply(e.y, function(E) bquote(10^(E)))))
par(op)

```

plot.design

[factoraov\(\)](#)

```
plot.design(x, y = NULL, fun = mean, data = NULL, ...,
            ylim = NULL, xlab = "Factors", ylab = NULL,
            main = NULL, ask = NULL, xaxt = par("xaxt"),
            axes = TRUE, xtick = FALSE)
```

x [formulateterms](#)

y

fun

data x

... [colpar](#)

ylim [plot.default](#)

xlab [title](#)

ylab

main [title](#)

ask

xaxt

axes

xtick

[fun\(\)](#)

[colfg](#)

[plotdesign](#)

[interaction.plot](#)

```

require(stats)
plot.design(warpbreaks) # automatic for data frame with one numeric var.

Form <- breaks ~ wool + tension
summary(fm1 <- aov(Form, data = warpbreaks))
plot.design(Form, data = warpbreaks, col = 2) # same as above

## More than one y :
utils::str(esoph)
plot.design(esoph) ## two plots; if interactive you are "ask"ed

## or rather, compare mean and median:
op <- par(mfcol = 1:2)
plot.design(ncases/ncontrols ~ ., data = esoph, ylim = c(0, 0.8))
plot.design(ncases/ncontrols ~ ., data = esoph, ylim = c(0, 0.8),
            fun = median)
par(op)

```

plot.factor

[factorplot](#)

[ybarplot](#)[yboxplot](#)[yspineplot](#)[yplot](#)[plot.default](#)

```

## S3 method for class 'factor'
plot(x, y, legend.text = NULL, ...)

```

```

xy          y
legend.text  ylevels(y)yaxlabelsspineplot
...         barplotboxplotspineplotplotpartitleaxes = FALSEtype

```

[plot.default](#)[plot.formula](#)[barplot](#)[boxplot](#)[spineplot](#)

```

require(grDevices)

plot(state.region)

## called from the formula method
plot(~ group, data = PlantGrowth)
plot(weight ~ group, data = PlantGrowth) # numeric ~ factor
plot(cut(weight, 2) ~ group, data = PlantGrowth) # factor ~ factor
## passing "..." to spineplot() eventually:
plot(cut(weight, 3) ~ group, data = PlantGrowth,
     col = hcl(c(0, 120, 240), 50, 70))

plot(PlantGrowth$group, axes = FALSE, main = "no axes") # extremely silly

```

plot.formula

```
## S3 method for class 'formula'
plot(formula, data = parent.frame(), ..., subset,
      ylab = varnames[response], ask = dev.interactive())
```

```
## S3 method for class 'formula'
points(formula, data = parent.frame(), ..., subset)
```

```
## S3 method for class 'formula'
lines(formula, data = parent.frame(), ..., subset)
```

```
## S3 method for class 'formula'
text(formula, data = parent.frame(), ..., subset)
```

```
formula      formulay ~ x
data          formula
...           horizontal = TRUE
subset
ylab
ask           par
```

```
linespointstexty ~ xy ~ 1plot
...dataparent.frame()data...datasubsetplot(y ~ 1)
plot
plot~ z + y + zplot.data.frame
```

```
yclassplot.formulaxhorizontal = TRUEboxplot
quotemainsubxlabpanel.firstpanel.lastplot.default
```

```
plot.defaultpointslinesplot.factor
```



```

op <- par(mfrow = c(2,1))
plot(Ozone ~ Wind, data = airquality, pch = as.character(Month))
plot(Ozone ~ Wind, data = airquality, pch = as.character(Month),
      subset = Month != 7)
par(op)

## text.formula() can be very natural:
wb <- within(warpbreaks, {
  time <- seq_along(breaks); W.T <- wool:tension })
plot(breaks ~ time, data = wb, type = "b")
text(breaks ~ time, data = wb, labels = W.T, col = 1+as.integer(wool))

```

plot.histogram

"histogram"[hist](#)

```

## S3 method for class 'histogram'
plot(x, freq = equidist, density = NULL, angle = 45,
      col = "lightgray", border = NULL, lty = NULL,
      main = paste("Histogram of", paste(x$xname, collapse = "\n")),
      sub = NULL, xlab = x$xname, ylab,
      xlim = range(x$breaks), ylim = NULL, log = "",
      axes = TRUE, labels = FALSE, add = FALSE,
      ann = TRUE, ...)

```

```

## S3 method for class 'histogram'
lines(x, ...)

```

x	histogram list densitymid hist x
freq	TRUEx\$countsFALSEx\$densitybreaks
col	NULL
border	
angledensity	rect
lty	lines
mainsubxlabylab	
	title
xlimylim	
log	""x"
axes	
labels	FALSETRUElabelscharacter
add	TRUElines.histogram(*)
ann	
...	titleaxis

```
lines.histogram(*)plot.histogram(*, add = TRUE)
```

histstemdensity

```
(wwt <- hist(women$weight, nclass = 7, plot = FALSE))
plot(wwt, labels = TRUE) # default main & xlab using wwt$xname
plot(wwt, border = "dark blue", col = "light blue",
     main = "Histogram of 15 women's weights", xlab = "weight [pounds]")

## Fake "lines" example, using non-default labels:
w2 <- wwt; w2$counts <- w2$counts - 1
lines(w2, col = "Midnight Blue", labels = ifelse(w2$counts, "> 1", "1"))
```

plot.raster

plot

```
## S3 method for class 'raster'
plot(x, y,
     xlim = c(0, ncol(x)), ylim = c(0, nrow(x)),
     xaxs = "i", yaxs = "i",
     asp = 1, add = FALSE, ...)

xy          y
xlimylim
xaxsyaxs
asp
add
...         rasterImage
```

plot.defaultrasterImage

```
require(grDevices)
r <- as.raster(c(0.5, 1, 0.5))
plot(r)
# additional arguments to rasterImage()
plot(r, interpolate=FALSE)
# distort
plot(r, asp=NA)
# fill page
```

```

op <- par(mar=rep(0, 4))
plot(r, asp=NA)
par(op)
# normal annotations work
plot(r, asp=NA)
box()
title(main="This is my raster")
# add to existing plot
plot(1)
plot(r, add=TRUE)

```

plot.table	table
------------	-------

plot

```

## S3 method for class 'table'
plot(x, type = "h", ylim = c(0, max(x)), lwd = 2,
      xlab = NULL, ylab = NULL, frame.plot = is.num, ...)
## S3 method for class 'table'
points(x, y = NULL, type = "h", lwd = 2, ...)
## S3 method for class 'table'
lines(x, y = NULL, type = "h", lwd = 2, ...)

```

x	table
y	NULL
type	
ylim	
lwd	type = "h"
xlabylab	
frame.plot	boxxdimnames
...	plot.defaultaxes = FALSE

plot.factorplot

```

## 1-d tables
(Poiss.tab <- table(N = stats::rpois(200, lambda = 5)))
plot(Poiss.tab, main = "plot(table(rpois(200, lambda = 5)))")

plot(table(state.division))

## 4-D :
plot(Titanic, main = "plot(Titanic, main= *)")

```

plot.window

[plot.default](#)[plot.new](#)

plot.window(xlim, ylim, log = "", asp = NA, ...)

xlimylim

log

asp

... [par](#)xaxsyaxslab

aspyasp $\times x$

[par](#)("usr")[par](#)("xaxs")asp

asp == 1asp > 1

xaxsyaxs"r"

xlimylimc(hi, lo)

xlimylim

[plot.hist](#)[image](#)

usrxaxpyaxplab

[xy.coords](#)[plot.xy](#)[plot.default](#)

[par](#)

```
##--- An example for the use of 'asp' :
require(stats) # normally loaded
loc <- cmdscale(eurodist)
rx <- range(x <- loc[,1])
ry <- range(y <- -loc[,2])
plot(x, y, type = "n", asp = 1, xlab = "", ylab = "")
abline(h = pretty(rx, 10), v = pretty(ry, 10), col = "lightgray")
text(x, y, labels(eurodist), cex = 0.8)
```

plot.xy

```
plot.xy(xy, type, pch = par("pch"), lty = par("lty"),
        col = par("col"), bg = NA,
        cex = 1, lwd = par("lwd"), ...)
```

xy	xy.coords
type	plot.default <code>NULL</code> "p"
pch	points.default
lty	lines
col	colourspalette <code>NULL</code>
bg	points.default
cex	
lwd	linespoints
...	<code>xpdlendljoinlmitre</code>

```
pchcolbgcexlwdtypepointslineslwd
cexpar("cex")type"p""o""b""c"
```

[plot](#)[plot.default](#)[points](#)[lines](#)

```
points.default # to see how it calls "plot.xy(xy.coords(x, y), ...)"
```

points

points

```
points(x, ...)
```

```
## Default S3 method:
points(x, y = NULL, type = "p", ...)
```

```
xy
type          typeplot.default
...
```

```
xyxy.coords
```

```
pch pch = 0:18
    pch = "."pch = 46cexcex = 1pdfpostscript
    cex = 1pointsizepch0:25par("cin")
```

```
col par
bg pch = 21:25
cex par("cex")
lwd par
ltylwd"b""l"
pchcolbgcexlwdtype = "b"lwd
xypchcolcexNA
```

```
pch
```

```
pch
    NA_integer_
    0:18
    19:25
    26:31
    32:127
    128:255128:159
    -32 ...
```

```
110131615:18
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
□	○	△	+	×	◇	▽	⊠	✱	⊕	⊗	⊞	⊠	⊠	⊠	■	●	▲	◆	●	●	●	◻	◇	△	▽

```
pch = 19:2521:25colbg
"grey"
```

```
pch = 19
pch = 2019
pch = 21
pch = 22
pch = 23
```

```
pch = 24
```

```
pch = 25
```

[pdf](#)

```
pch = 16pch = 19lwdcex
```

```
pch = 26:31pch = 32:127pch = 128:255-0x2642L-0x20ACL
```

```
32:127128:255
```

```
pchNA
```

```
pchNULLpar("pch")
```

```
par(font = 5)pchc(32:126, 160:254)240160
```

```
pch = 128:255
```

```
cex
```

[points.formulaplotlinesplot.xy](#)

```
require(stats) # for rnorm
```

```
plot(-4:4, -4:4, type = "n") # setting up coord. system
```

```
points(rnorm(200), rnorm(200), col = "red")
```

```
points(rnorm(100)/2, rnorm(100)/2, col = "blue", cex = 1.5)
```

```
op <- par(bg = "light blue")
```

```
x <- seq(0, 2*pi, length.out = 51)
```

```
## something "between type='b' and type='o'":
```

```
plot(x, sin(x), type = "o", pch = 21, bg = par("bg"), col = "blue", cex = .6,
```

```
main = 'plot(..., type="o", pch=21, bg=par("bg"))')
```

```
par(op)
```

```
## Illustration of pch = 0:25 (as in the figure shown above in PDF/HTML help)
```

```
## Not run: png("pch.png", height = 0.7, width = 7, res = 100, units = "in")
```

```
par(mar = rep(0,4))
```

```
plot(c(-1, 26), 0:1, type = "n", axes = FALSE)
```

```
text(0:25, 0.6, 0:25, cex = 0.5)
```

```
points(0:25, rep(0.3, 26), pch = 0:25, bg = "grey")
```

```
##----- Showing all the extra & some char graphics symbols -----
```

```
pchShow <-
```

```
function(extras = c("*", ".", "o", "O", "0", "+", "-", "|", "%", "#"),
  cex = 3, ## good for both .Device=="postscript" and "x11"
  col = "red3", bg = "gold", coltext = "brown", cextext = 1.2,
  main = paste("plot symbols : points (... pch = *, cex =",
    cex, ")"))
```

```
{
```

```

nex <- length(extras)
np <- 26 + nex
ipch <- 0:(np-1)
k <- floor(sqrt(np))
dd <- c(-1,1)/2
rx <- dd + range(ix <- ipch %% k)
ry <- dd + range(iy <- 3 + (k-1)- ipch %% k)
pch <- as.list(ipch) # list with integers & strings
if(nex > 0) pch[26+ 1:nex] <- as.list(extras)
plot(rx, ry, type = "n", axes = FALSE, xlab = "", ylab = "", main = main)
abline(v = ix, h = iy, col = "lightgray", lty = "dotted")
for(i in 1:np) {
  pc <- pch[[i]]
  ## 'col' symbols with a 'bg'-colored interior (where available) :
  points(ix[i], iy[i], pch = pc, col = col, bg = bg, cex = cex)
  if(cextext > 0)
    text(ix[i] - 0.3, iy[i], pc, col = coltext, cex = cextext)
}
}

pchShow()
pchShow(c("o","0","0"), cex = 2.5)
pchShow(NULL, cex = 4, cextext = 0, main = NULL)

## ----- test code for various pch specifications -----
# Try this in various font families (including Hershey)
# and locales. Use sign = -1 asserts we want Latin-1.
# Standard cases in a MBCS locale will not plot the top half.
TestChars <- function(sign = 1, font = 1, ...)
{
  MB <- l10n_info()$MBCS
  r <- if(font == 5) { sign <- 1; c(32:126, 160:254)
  } else if(MB) 32:126 else 32:255
  if (sign == -1) r <- c(32:126, 160:255)
  par(pty = "s")
  plot(c(-1,16), c(-1,16), type = "n", xlab = "", ylab = "",
        xaxs = "i", yaxs = "i",
        main = sprintf("sign = %d, font = %d", sign, font))
  grid(17, 17, lty = 1) ; mtext(paste("MBCS:", MB))
  for(i in r) try(points(i%%16, i%%16, pch = sign*i, font = font,...))
}
TestChars()
try(TestChars(sign = -1))
TestChars(font = 5) # Euro might be at 160 (0+10*16).
# macOS has apple at 240 (0+15*16).
try(TestChars(-1, font = 2)) # bold

```

polygon

polygonxy


```

polygon(x, y = NULL, density = NULL, angle = 45,
        border = NULL, col = NA, lty = par("lty"),
        ..., fillOddEven = FALSE)

```

```

xy
density          NULLdensityNA
angle
col              NAdensityNULLNAdensity
border          NULLpar("fg")border = NA
                borderFALSENATRUENULL
lty              par
...              xpdlendljoinlmitre
fillOddEven      FALSE

```

```

xyxy.coords

```

```

linesNA
densityanglecolborderlty

```

```

fillOddEvenwindowspdfpostscriptfillOddEven

```

```

<buhr@stat.wisc.edu>

```

```

segmentslinesrectboxabline
par

```

```

x <- c(1:9, 8:1)
y <- c(1, 2*(5:3), 2, -1, 17, 9, 8, 2:9)
op <- par(mfcol = c(3, 1))
for(xpd in c(FALSE, TRUE, NA)) {
  plot(1:10, main = paste("xpd =", xpd))
  box("figure", col = "pink", lwd = 3)
  polygon(x, y, xpd = xpd, col = "orange", lty = 2, lwd = 2, border = "red")
}
par(op)

```

```

n <- 100
xx <- c(0:n, n:0)
yy <- c(c(0, cumsum(stats::rnorm(n))), rev(c(0, cumsum(stats::rnorm(n)))))
plot (xx, yy, type = "n", xlab = "Time", ylab = "Distance")
polygon(xx, yy, col = "gray", border = "red")
title("Distance Between Brownian Motions")

# Multiple polygons from NA values
# and recycling of col, border, and lty
op <- par(mfrow = c(2, 1))
plot(c(1, 9), 1:2, type = "n")
polygon(1:9, c(2,1,2,1,1,2,1,2,1),
        col = c("red", "blue"),
        border = c("green", "yellow"),
        lwd = 3, lty = c("dashed", "solid"))
plot(c(1, 9), 1:2, type = "n")
polygon(1:9, c(2,1,2,1,NA,2,1,2,1),
        col = c("red", "blue"),
        border = c("green", "yellow"),
        lwd = 3, lty = c("dashed", "solid"))
par(op)

# Line-shaded polygons
plot(c(1, 9), 1:2, type = "n")
polygon(1:9, c(2,1,2,1,NA,2,1,2,1),
        density = c(10, 20), angle = c(-45, 45))

```

polypath

pathxy

```

polypath(x, y = NULL,
         border = NULL, col = NA, lty = par("lty"),
         rule = "winding", ...)

```

xy	
col	NA
border	NULLpar("fg")border = NA borderFALSENATRUENULL
lty	par
rule	"winding""evenodd"
...	xpdlendljoinlmitre

xyxy.coords

polygonNA

polygon()

xfigpictex

segmentslinesrectboxpolygon

par

```
plotPath <- function(x, y, col = "grey", rule = "winding") {
  plot.new()
  plot.window(range(x, na.rm = TRUE), range(y, na.rm = TRUE))
  polypath(x, y, col = col, rule = rule)
  if (!is.na(col))
    mtext(paste("Rule:", rule), side = 1, line = 0)
}

plotRules <- function(x, y, title) {
  plotPath(x, y)
  plotPath(x, y, rule = "evenodd")
  mtext(title, side = 3, line = 0)
  plotPath(x, y, col = NA)
}

op <- par(mfrow = c(5, 3), mar = c(2, 1, 1, 1))

plotRules(c(.1, .1, .9, .9, NA, .2, .2, .8, .8),
  c(.1, .9, .9, .1, NA, .2, .8, .8, .2),
  "Nested rectangles, both clockwise")
plotRules(c(.1, .1, .9, .9, NA, .2, .8, .8, .2),
  c(.1, .9, .9, .1, NA, .2, .2, .8, .8),
  "Nested rectangles, outer clockwise, inner anti-clockwise")
plotRules(c(.1, .1, .4, .4, NA, .6, .9, .9, .6),
  c(.1, .4, .4, .1, NA, .6, .6, .9, .9),
  "Disjoint rectangles")
plotRules(c(.1, .1, .6, .6, NA, .4, .4, .9, .9),
  c(.1, .6, .6, .1, NA, .4, .9, .9, .4),
  "Overlapping rectangles, both clockwise")
plotRules(c(.1, .1, .6, .6, NA, .4, .9, .9, .4),
  c(.1, .6, .6, .1, NA, .4, .4, .9, .9),
  "Overlapping rectangles, one clockwise, other anti-clockwise")

par(op)
```

rasterImage

rasterImage

```
rasterImage(image,  
            xleft, ybottom, xright, ytop,  
            angle = 0, interpolate = TRUE, ...)
```

image [raster](#)[as.raster](#)

xleft

ybottom

xright

ytop

angle

interpolate

...

xleft, ...xleftxright

postscriptX11(type = "Xlib")windows()

[rectpolygonsegments](#)

[dev.capabilities](#)

```
require(grDevices)  
## set up the plot region:  
op <- par(bg = "thistle")  
plot(c(100, 250), c(300, 450), type = "n", xlab = "", ylab = "")  
image <- as.raster(matrix(0:1, ncol = 5, nrow = 3))  
rasterImage(image, 100, 300, 150, 350, interpolate = FALSE)  
rasterImage(image, 100, 400, 150, 450)  
rasterImage(image, 200, 300, 200 + xinch(.5), 300 + yinch(.3),  
            interpolate = FALSE)  
rasterImage(image, 200, 400, 250, 450, angle = 15, interpolate = FALSE)  
par(op)
```

rect

rect

```
rect(xleft, ybottom, xright, ytop, density = NULL, angle = 45,
     col = NA, border = NULL, lty = par("lty"), lwd = par("lwd"),
     ...)
```

xleft

ybottom

xright

ytop

density NULLdensityNA

angle

col NANULLdensity

border par("fg")border = NAborder = TRUE

lty "solid"

lwd lwd = 0

... xpdlendljoinlmitre

xleft, ...xleftxright

[histbarplotlegend](#)

[boxpolygonsegments](#)

[par](#)

```
require(grDevices)
## set up the plot region:
op <- par(bg = "thistle")
plot(c(100, 250), c(300, 450), type = "n", xlab = "", ylab = "",
     main = "2 x 11 rectangles; 'rect(100+i,300+i, 150+i,380+i)'" )
i <- 4*(0:10)
## draw rectangles with bottom left (100, 300)+i
## and top right (150, 380)+i
rect(100+i, 300+i, 150+i, 380+i, col = rainbow(11, start = 0.7, end = 0.1))
rect(240-i, 320+i, 250-i, 410+i, col = heat.colors(11), lwd = i/5)
## Background alternating ( transparent / "bg" ) :
j <- 10*(0:5)
rect(125+j, 360+j, 141+j, 405+j/2, col = c(NA,0),
     border = "gold", lwd = 2)
rect(125+j, 296+j/2, 141+j, 331+j/5, col = c(NA,"midnightblue"))
```

```

mtext("+ 2 x 6 rect(*, col = c(NA,0)) and col = c(NA,\"m..blue\")")

## an example showing colouring and shading
plot(c(100, 200), c(300, 450), type= "n", xlab = "", ylab = "")
rect(100, 300, 125, 350) # transparent
rect(100, 400, 125, 450, col = "green", border = "blue") # coloured
rect(115, 375, 150, 425, col = par("bg"), border = "transparent")
rect(150, 300, 175, 350, density = 10, border = "red")
rect(150, 400, 175, 450, density = 30, col = "blue",
      angle = -30, border = "transparent")

legend(180, 450, legend = 1:4, fill = c(NA, "green", par("fg"), "blue"),
       density = c(NA, NA, 10, 30), angle = c(NA, NA, 30, -30))

par(op)

```

rug

```

rug(x, ticksize = 0.03, side = 1, lwd = 0.5, col = par("fg"),
    quiet = getOption("warn") < 0, ...)

```

```

x
ticksize
side
lwd          1
col
quiet
...          axislinepos

```

rugx

[jitterx](#)

```

require(stats) # both 'density' and its default method
with(faithful, {
  plot(density(eruptions, bw = 0.15))
  rug(eruptions)
  rug(jitter(eruptions, amount = 0.01), side = 3, col = "light blue")
})

```

screen

split.screen

screen

erase.screen

close.screen

split.screen(figs, screen, erase = TRUE)

screen(n = , new = TRUE)

erase.screen(n =)

close.screen(n, all.screens = FALSE)

figs

screen

erase

n screenerase.screenclose.screenclose.screen

new

all.screens

split.screenclose.screen(all = TRUE)

close.screen

split.screen(*)split.screen()

screen(n)nscreen()

close.screen()

screenerase.screenclose.screenFALSE

[par](#)(mfrow)[par](#)(mfcol)[layout](#)()

[coplot](#)

erase.screen

[parlayoutDevices](#)dev.*

```

if (interactive()) {
  par(bg = "white")      # default is likely to be transparent
  split.screen(c(2, 1))  # split display into two screens
  split.screen(c(1, 3), screen = 2) # now split the bottom half into 3
  screen(1) # prepare screen 1 for output
  plot(10:1)
  screen(4) # prepare screen 4 for output
  plot(10:1)
  close.screen(all = TRUE) # exit split-screen mode

  split.screen(c(2, 1))  # split display into two screens
  split.screen(c(1, 2), 2) # split bottom half in two
  plot(1:10)             # screen 3 is active, draw plot
  erase.screen()          # forgot label, erase and redraw
  plot(1:10, ylab = "ylab 3")
  screen(1)              # prepare screen 1 for output
  plot(1:10)
  screen(4)              # prepare screen 4 for output
  plot(1:10, ylab = "ylab 4")
  screen(1, FALSE)       # return to screen 1, but do not clear
  plot(10:1, axes = FALSE, lty = 2, ylab = "") # overlay second plot
  axis(4)                # add tic marks to right-hand axis
  title("Plot 1")
  close.screen(all = TRUE) # exit split-screen mode
}

```

segments

```

segments(x0, y0, x1 = x0, y1 = y0,
         col = par("fg"), lty = par("lty"), lwd = par("lwd"),
         ...)

```

```

x0y0
x1y1
colltylwd      parNAcol
...            parxpdlendljoinlmitre

```

```

i(x0[i], y0[i])(x1[i], y1[i])
colltylwd

```


arrowspolygonlines

```
x <- stats::runif(12); y <- stats::rnorm(12)
i <- order(x, y); x <- x[i]; y <- y[i]
plot(x, y, main = "arrows(.) and segments(.)")
## draw arrows from point to point :
s <- seq(length(x)-1) # one shorter than data
arrows(x[s], y[s], x[s+1], y[s+1], col= 1:3)
s <- s[-length(s)]
segments(x[s], y[s], x[s+2], y[s+2], col= 'pink')
```

smoothScatter

smoothScatter

```
smoothScatter(x, y = NULL, nbin = 128, bandwidth,
              colramp = colorRampPalette(c("white", blues9)),
              nrpoints = 100, ret.selection = FALSE,
              pch = ".", cex = 1, col = "black",
              transformation = function(x) x^.25,
              postPlotHook = box,
              xlab = NULL, ylab = NULL, xlim, ylim,
              xaxs = par("xaxs"), yaxs = par("yaxs"), ...)
```

xy	xy xy.coords
nbin	gridsize bkde2D()
bandwidth	bandwidth bkde2D
colramp	nn
nrpoints	nrpoints nrpoints = Inf
ret.selection	logical nrpoints > 0
pchcexcol	points nrpoints > 0 par
transformation	
postPlotHook	NULL image
xlaby	lab
ylim	
xaxsyaxs...	image add=TRUE useRaster=TRUE

[smoothScatter](#)[bkde2D](#)[pairs](#)

[ret.selection](#)[nrpoints](#)[xlim](#)[ylim](#)

[bkde2DdensColsblues9](#)

[scatter.smoothloess](#)

```
## A largish data set
n <- 10000
x1 <- matrix(rnorm(n), ncol = 2)
x2 <- matrix(rnorm(n, mean = 3, sd = 1.5), ncol = 2)
x <- rbind(x1, x2)

oldpar <- par(mfrow = c(2, 2), mar=.1+c(3,3,1,1), mgp = c(1.5, 0.5, 0))
smoothScatter(x, nrpoints = 0)
smoothScatter(x)

## a different color scheme:
Lab.palette <- colorRampPalette(c("blue", "orange", "red"), space = "Lab")
i.s <- smoothScatter(x, colramp = Lab.palette,
                     ## pch=NA: do not draw them
                     nrpoints = 250, ret.selection=TRUE)
## label the 20 very lowest-density points, the "outliers" (with obs.number):
i.20 <- i.s[1:20]
text(x[i.20,], labels = i.20, cex= 0.75)

## somewhat similar, using identical smoothing computations,
## but considerably *less* efficient for really large data:
plot(x, col = densCols(x), pch = 20)

## use with pairs:
par(mfrow = c(1, 1))
y <- matrix(rnorm(40000), ncol = 4) + 3*rnorm(10000)
y[, c(2,4)] <- -y[, c(2,4)]
pairs(y, panel = function(...) smoothScatter(..., nrpoints = 0, add = TRUE),
      gap = 0.2)

par(oldpar)
```

spineplot

spineplot(x, ...)

Default S3 method:

```
spineplot(x, y = NULL,
          breaks = NULL, tol.ylab = 0.05, off = NULL,
```

```

      ylevels = NULL, col = NULL,
      main = "", xlab = NULL, ylab = NULL,
      xaxlabels = NULL, yaxlabels = NULL,
      xlim = NULL, ylim = c(0, 1), axes = TRUE, weights = NULL, ...)

## S3 method for class 'formula'
spineplot(formula, data = NULL,
           breaks = NULL, tol.ylab = 0.05, off = NULL,
           ylevels = NULL, col = NULL,
           main = "", xlab = NULL, ylab = NULL,
           xaxlabels = NULL, yaxlabels = NULL,
           xlim = NULL, ylim = c(0, 1), axes = TRUE, ...,
           subset = NULL, weights = NULL, drop.unused.levels = FALSE)

x
y          "factor"
formula    "formula"y ~ x"factor"
data
breaks     breakshist
tol.ylab
off        02
ylevels
col        levels(y)gray.colors
mainxlabylab
xaxlabelsyaxlabels
           levels(y)levels(x)xaxlabels
xlimylim
axes       FALSE
weights    NULLx
...        rect
subset
drop.unused.levels
           FALSE

spineplotspineplot(x, y)spineplot(y ~ x)yxxspineplottable(x, y)
P(y|x)P(x)xyxxhistbreaks
xyx

```

<Achim.Zeileis@R-project.org>

mosaicplot histcdplot

```
## treatment and improvement of patients with rheumatoid arthritis
treatment <- factor(rep(c(1, 2), c(43, 41)), levels = c(1, 2),
                    labels = c("placebo", "treated"))
improved <- factor(rep(c(1, 2, 3, 1, 2, 3), c(29, 7, 7, 13, 7, 21)),
                   levels = c(1, 2, 3),
                   labels = c("none", "some", "marked"))

## (dependence on a categorical variable)
(spineplot(improved ~ treatment))

## applications and admissions by department at UC Berkeley
## (two-way tables)
(spineplot(marginSums(UCBAdmissions, c(3, 2)),
           main = "Applications at UCB"))
(spineplot(marginSums(UCBAdmissions, c(3, 1)),
           main = "Admissions at UCB"))

## NASA space shuttle o-ring failures
fail <- factor(c(2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1,
                 1, 1, 1, 2, 1, 1, 1, 1, 1),
               levels = c(1, 2), labels = c("no", "yes"))
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68, 69, 70, 70,
                 70, 70, 72, 73, 75, 75, 76, 76, 78, 79, 81)

## (dependence on a numerical variable)
(spineplot(fail ~ temperature))
(spineplot(fail ~ temperature, breaks = 3))
(spineplot(fail ~ temperature, breaks = quantile(temperature)))

## highlighting for failures
spineplot(fail ~ temperature, ylevels = 2:1)
```

```

stars(x, full = TRUE, scale = TRUE, radius = TRUE,
      labels = dimnames(x)[[1]], locations = NULL,
      nrow = NULL, ncol = NULL, len = 1,
      key.loc = NULL, key.labels = dimnames(x)[[2]],
      key.xpd = TRUE,
      xlim = NULL, ylim = NULL, flip.labels = NULL,
      draw.segments = FALSE,
      col.segments = 1:n.seg, col.stars = NA, col.lines = NA,
      axes = FALSE, frame.plot = axes,
      main = NULL, sub = NULL, xlab = "", ylab = "",
      cex = 0.8, lwd = 0.25, lty = par("lty"), xpd = FALSE,
      mar = pmin(par("mar"),
                  1.1+ c(2*axes+ (xlab != ""),
                        2*axes+ (ylab != ""), 1, 0)),
      add = FALSE, plot = TRUE, ...)

```

x	xNA
full	TRUE
scale	TRUEFALSE[0,1]
radius	TRUE
labels	starslabels = NULL
locations	locations = NULL
nrowncol	locationsNULLnrow == ncol
len	
key.loc	
key.labels	dimnames(x)
key.xpd	par("xpd")
xlim	
ylim	
flip.labels	
draw.segments	TRUE
col.segments	pardraw.segments = FALSE
col.stars	pardraw.segments = TRUE
col.lines	pardraw.segments = TRUE
axes	TRUE
frame.plot	TRUE
main	
sub	
xlab	
ylab	
cex	
lwd	

```

lty
xpd          par(xpd = .)
mar          par(mar = *)
...          plot()plot.defaultbox()frame.plot
add          TRUE
plot         FALSE

```

x

plot = TRUE

symbols

```

require(grDevices)
stars(mtcars[, 1:7], key.loc = c(14, 2),
      main = "Motor Trend Cars : stars(*, full = F)", full = FALSE)
stars(mtcars[, 1:7], key.loc = c(14, 1.5),
      main = "Motor Trend Cars : full stars()", flip.labels = FALSE)

## 'Spider' or 'Radar' plot:
stars(mtcars[, 1:7], locations = c(0, 0), radius = FALSE,
      key.loc = c(0, 0), main = "Motor Trend Cars", lty = 2)

## Segment Diagrams:
palette(rainbow(12, s = 0.6, v = 0.75))
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),
      main = "Motor Trend Cars", draw.segments = TRUE)
stars(mtcars[, 1:7], len = 0.6, key.loc = c(1.5, 0),
      main = "Motor Trend Cars", draw.segments = TRUE,
      frame.plot = TRUE, nrow = 4, cex = .7)

## scale linearly (not affinely) to [0, 1]
USJudge <- apply(USJudgeRatings, 2, function(x) x/max(x))
Jnam <- row.names(USJudgeRatings)

```

```

Snam <- abbreviate(substring(Jnam, 1, regexpr("[.]", Jnam) - 1), 7)
stars(USJudge, labels = Jnam, scale = FALSE,
      key.loc = c(13, 1.5), main = "Judge not ...", len = 0.8)
stars(USJudge, labels = Snam, scale = FALSE,
      key.loc = c(13, 1.5), radius = FALSE)

loc <- stars(USJudge, labels = NULL, scale = FALSE,
            radius = FALSE, frame.plot = TRUE,
            key.loc = c(13, 1.5), main = "Judge not ...", len = 1.2)
text(loc, Snam, col = "blue", cex = 0.8, xpd = TRUE)

## 'Segments':
stars(USJudge, draw.segments = TRUE, scale = FALSE, key.loc = c(13,1.5))

## 'Spider':
stars(USJudgeRatings, locations = c(0, 0), scale = FALSE, radius = FALSE,
      col.stars = 1:10, key.loc = c(0, 0), main = "US Judges rated")
## Same as above, but with colored lines instead of filled polygons.
stars(USJudgeRatings, locations = c(0, 0), scale = FALSE, radius = FALSE,
      col.lines = 1:10, key.loc = c(0, 0), main = "US Judges rated")
## 'Radar-Segments'
stars(USJudgeRatings[1:10,], locations = 0:1, scale = FALSE,
      draw.segments = TRUE, col.segments = 0, col.stars = 1:10, key.loc = 0:1,
      main = "US Judges 1-10 ")
palette("default")
stars(cbind(1:16, 10*(16:1)), draw.segments = TRUE,
      main = "A Joke -- do *not* use symbols on 2D data!")

```

stem

```
stemxscale = 2
```

```
stem(x, scale = 1, width = 80, atom = 1e-08)
```

```

x
scale
width
atom

```

```
x
```

```

stem(islands)
stem(log10(islands))

```

stripchart

stripchart[boxplot](#)

stripchart(x, ...)

```
## S3 method for class 'formula'
stripchart(x, data = NULL, dlab = NULL, ...,
           subset, na.action = NULL)
```

```
## Default S3 method:
stripchart(x, method = "overplot", jitter = 0.1, offset = 1/3,
           vertical = FALSE, group.names, add = FALSE,
           at = NULL, xlim = NULL, ylim = NULL,
           ylab = NULL, xlab = NULL, dlab = "", glab = "",
           log = "", pch = 0, col = par("fg"), cex = par("cex"),
           axes = TRUE, frame.plot = axes, ...)
```

x	formulay ~ gygNA
data	x
subset	
na.action	NA
...	plot.window points xaxis title
method	"overplot""jitter""stack"
jitter	method = "jitter"jitter
offset	
vertical	TRUE
group.names	
add	
at	add = TRUE1:nn
ylabxlab	title
dlabglab	
xlimylim	plot.window
log	plot.default
pchcolcex	par
axesframe.plot	plot.default

dlabglabxlabylabdlabverticalTRUEglab


```

x <- stats::rnorm(50)
xr <- round(x, 1)
stripchart(x) ; m <- mean(par("usr")[1:2])
text(m, 1.04, "stripchart(x, \"overplot\")")
stripchart(xr, method = "stack", add = TRUE, at = 1.2)
text(m, 1.35, "stripchart(round(x,1), \"stack\")")
stripchart(xr, method = "jitter", add = TRUE, at = 0.7)
text(m, 0.85, "stripchart(round(x,1), \"jitter\")")

stripchart(decrease ~ treatment,
  main = "stripchart(OrchardSprays)",
  vertical = TRUE, log = "y", data = OrchardSprays)

stripchart(decrease ~ treatment, at = c(1:8)^2,
  main = "stripchart(OrchardSprays)",
  vertical = TRUE, log = "y", data = OrchardSprays)

```

strwidth

```
s[i]par("fin")
```

```

strwidth(s, units = "user", cex = NULL, font = NULL, vfont = NULL, ...)
strheight(s, units = "user", cex = NULL, font = NULL, vfont = NULL, ...)

```

```

s                as.graphicsAnnot
units            s"user""inches""figure"
cex              par("cex")NULL1
fontvfont...    "family"text

```

```

"\n""M"cexpar("lheight")
"user"plot.new

```

```
ss[i]NA
```

textnchar

```

str.ex <- c("W","w","I",".", "WwI.")
op <- par(pty = "s"); plot(1:100, 1:100, type = "n")
sw <- strwidth(str.ex); sw
all.equal(sum(sw[1:4]), sw[5])
#- since the last string contains the others

sw.i <- strwidth(str.ex, "inches"); 25.4 * sw.i # width in [mm]
unique(sw / sw.i)
# constant factor: 1 value
mean(sw.i / strwidth(str.ex, "fig")) / par('fin')[1] # = 1: are the same

## See how letters fall in classes
## -- depending on graphics device and font!
all.lett <- c(letters, LETTERS)
shL <- strheight(all.lett, units = "inches") * 72 # 'big points'
table(shL) # all have same heights ...
mean(shL)/par("cin")[2] # around 0.6

(swL <- strwidth(all.lett, units = "inches") * 72) # 'big points'
split(all.lett, factor(round(swL, 2)))

sumex <- expression(sum(x[i], i=1,n), e^{i * pi} == -1)
strwidth(sumex)
strheight(sumex)

par(op) #- reset to previous setting

```

sunflowerplot

```

sunflowerplot(x, ...)

## Default S3 method:
sunflowerplot(x, y = NULL, number, log = "", digits = 6,
              xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
              add = FALSE, rotate = FALSE,
              pch = 16, cex = 0.8, cex.fact = 1.5,
              col = par("col"), bg = NA, size = 1/8, seg.col = 2,
              seg.lwd = 1.5, ...)

## S3 method for class 'formula'
sunflowerplot(formula, data = NULL, xlab = NULL, ylab = NULL, ...,
              subset, na.action = NULL)

x          xnplot.defaultxy.coords
y          yn

```

number	nnumber[i](x[i], y[i]) missing(number)x[], y[]xyTable()
log	plot.default
digits	numberxydigits
xlabylab	
xlimylim	numeric(2)
add	FALSE
rotate	TRUE
pch	number[i]==1
cex	pch
cex.fact	cex / cex.fact
colbg	plot.default
size	
seg.col	par(col=)col = "gold"
seg.lwd	
...	plotadd = FALSE
formula	formulay ~ x
data	formula
subset	
na.action	NA

```

number[i] == 1pchnumber[i] > 1number[i]
rotate = TRUEnumber[i] >= 2jitter

```

```

x
y
number
xyTable()

```

<maechler@stat.math.ethz.ch>

[densityxyTables](#)[smoothScatter\(\)](#)[sunflowerplot\(\)](#)

```
require(stats) # for rnorm
require(grDevices)

## 'number' is computed automatically:
sunflowerplot(iris[, 3:4])
## Imitating Chambers et al, p.109, closely:
sunflowerplot(iris[, 3:4], cex = .2, cex.fact = 1, size = .035, seg.lwd = .8)
## or
sunflowerplot(Petal.Width ~ Petal.Length, data = iris,
               cex = .2, cex.fact = 1, size = .035, seg.lwd = .8)

sunflowerplot(x = sort(2*round(rnorm(100))), y = round(rnorm(100), 0),
               main = "Sunflower Plot of Rounded N(0,1)")
## Similarly using a "xyTable" argument:
xyT <- xyTable(x = sort(2*round(rnorm(100))), y = round(rnorm(100), 0),
               digits = 3)
utils::str(xyT, vec.len = 20)
sunflowerplot(xyT, main = "2nd Sunflower Plot of Rounded N(0,1)")

## A 'marked point process' {explicit 'number' argument}:
sunflowerplot(rnorm(100), rnorm(100), number = rpois(n = 100, lambda = 2),
               main = "Sunflower plot (marked point process)",
               rotate = TRUE, col = "blue4")
```

symbols

```
symbols(x, y = NULL, circles, squares, rectangles, stars,
        thermometers, boxplots, inches = TRUE, add = FALSE,
        fg = par("col"), bg = NA,
        xlab = NULL, ylab = NULL, main = NULL,
        xlim = NULL, ylim = NULL, ...)
```

xy [xy.coords](#)

circles

squares

rectangles

stars NA

thermometers fgfgbg

boxplots

```

inches      TRUEFALSE
add         addTRUE
fg
bg          bg
xlab        adddeparsex
ylab        add = TRUE
main        add = TRUE
xlim        add = TRUE
ylim        add = TRUE
...         aspplot.window

```

```

NA
inchesTRUETRUE1inchesFALSEinches
windows

```

stars

```
symbols(*, circles=*)sunflowerplot
```

```

require(stats); require(grDevices)
x <- 1:10
y <- sort(10*runif(10))
z <- runif(10)
z3 <- cbind(z, 2*runif(10), runif(10))
symbols(x, y, thermometers = cbind(.5, 1, z), inches = .5, fg = 1:10)
symbols(x, y, thermometers = z3, inches = FALSE)
text(x, y, apply(format(round(z3, digits = 2)), 1, paste, collapse = ","),
     adj = c(-.2,0), cex = .75, col = "purple", xpd = NA)

## Note that example(trees) shows more sensible plots!
N <- nrow(trees)
with(trees, {
  ## Girth is diameter in inches
  symbols(Height, Volume, circles = Girth/24, inches = FALSE,
         main = "Trees' Girth") # xlab and ylab automatically
  ## Colours too:
  op <- palette(rainbow(N, end = 0.9))
  symbols(Height, Volume, circles = Girth/16, inches = FALSE, bg = 1:N,
         fg = "gray30", main = "symbols(*, circles = Girth/16, bg = 1:N)")
  palette(op)
})

```

text

textlabelsxyxy.coords(x, y)

text(x, ...)

Default S3 method:

```
text(x, y = NULL, labels = seq_along(x$x), adj = NULL,
      pos = NULL, offset = 0.5, vfont = NULL,
      cex = 1, col = NULL, font = NULL, ...)
```

xy	labelsxy
labels	as.characterlabelsxylabels
adj	[0, 1][0, 1]
pos	adj1234(x, y)
offset	pos
vfont	NULLHersheylabels
cex	par("cex")NULLNA1.0
colfont	vfont = NULLpar()
...	parsrtrfamilyxpd

labelscharacterexpression

```
adj(x, y)(x, y)Napar(adj)adj = c(par("adj"), NA)xlength(adj) == 1Ladjadj = c(adj,
NA)adj = c(0, 0)
```

posoffsetidentify

```
srtparadsrt(x, y)possrt(x, y)pos = 1pos = 2pos = 3pos = 4pos(x, y)adsrt
```

```
colcexfontlabelsNAfontpar("font")col
```

xylabelsNA

```
font = 5labels"d""\300"
```

```
text(x, y, "\xA0", font = 5)pdfpostscript\u20ac
```

```
X11postscriptpdfISOLatin9.enc
```

text.formulamtexttitleHersheyplotmath

```

plot(-1:1, -1:1, type = "n", xlab = "Re", ylab = "Im")
K <- 16; text(exp(1i * 2 * pi * (1:K) / K), col = 2)

## The following two examples use latin1 characters: these may not
## appear correctly (or be omitted entirely).
plot(1:10, 1:10, main = "text(...) examples\n~~~~~",
     sub = "R is GNU ©, but not ® ...")
mtext("«Latin-1 accented chars»: éè øð å&laquo; æ<Æ", side = 3)
points(c(6,2), c(2,1), pch = 3, cex = 4, col = "red")
text(6, 2, "the text is CENTERED around (x,y) = (6,2) by default",
     cex = .8)
text(2, 1, "or Left/Bottom - JUSTIFIED at (2,1) by 'adj = c(0,0)'",
     adj = c(0,0))
text(4, 9, expression(hat(beta) == (X^t * X)^{-1} * X^t * y))
text(4, 8.4, "expression(hat(beta) == (X^t * X)^{-1} * X^t * y)",
     cex = .75)
text(4, 7, expression(bar(x) == sum(frac(x[i], n), i==1, n)))

## Two more latin1 examples
text(5, 10.2,
     "Le français, c'est facile: Règles, Liberté, Egalité, Fraternité...")
text(5, 9.8,
     "Jetzt no chli züritüütsch: (noch ein bißchen Zürcher deutsch)")

```

title

characterexpression

```

title(main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
      line = NA, outer = FALSE, ...)

```

```

main      par(c("font.main", "cex.main", "col.main"))
sub       par(c("font.sub", "cex.sub", "col.sub"))
xlab      par(c("font.lab", "cex.lab", "col.lab"))
ylab      xlab
line      line
outer     TRUE
...       parcol.maincex.subcolcexadjxpdouter = TRUEmgp

```

```

titlecex=col=font=as.graphicsAnnot

```

```

mainpar("adj")outer = TRUE

```

```

xlabylabsublinexlabylabpar("mgp")[1]1subpar("mex")par("adj")

```

mtexttextplotmath

```
plot(cars, main = "") # here, could use main directly
title(main = "Stopping Distance versus Speed")

plot(cars, main = "")
title(main = list("Stopping Distance versus Speed", cex = 1.5,
                 col = "red", font = 3))

## Specifying "...":
plot(1, col.axis = "sky blue", col.lab = "thistle")
title("Main Title", sub = "subtitle",
      cex.main = 2, font.main = 4, col.main = "blue",
      cex.sub = 0.75, font.sub = 3, col.sub = "red")

x <- seq(-4, 4, length.out = 101)
y <- cbind(sin(x), cos(x))
matplot(x, y, type = "l", xaxt = "n",
        main = expression(paste(plain(sin) * phi, " and ",
                                plain(cos) * phi)),
        ylab = expression("sin" * phi, "cos" * phi), # only 1st is taken
        xlab = expression(paste("Phase Angle ", phi)),
        col.main = "blue")
axis(1, at = c(-pi, -pi/2, 0, pi/2, pi),
     labels = expression(-pi, -pi/2, 0, pi/2, pi))
abline(h = 0, v = pi/2 * c(-1,1), lty = 2, lwd = .1, col = "gray70")
```

units

xinchyinchloglogpar

xyinchxy

```
xinch(x = 1, warn.log = TRUE)
yinch(y = 1, warn.log = TRUE)
xyinch(xy = 1, warn.log = TRUE)
```

xy

xy

warn.log TRUE


```

all(c(xinch(), yinch()) == xyinch()) # TRUE
xyinch()
xyinch #- to see that is really  delta{"usr"} / "pin"

## plot labels offset 0.12 inches to the right
## of plotted symbols in a plot
with(mtcars, {
  plot(mpg, disp, pch = 19, main = "Motor Trend Cars")
  text(mpg + xinch(0.12), disp, row.names(mtcars),
       adj = 0, cex = .7, col = "blue")
})

```

xspline

```

xspline(x, y = NULL, shape = 0, open = TRUE, repEnds = TRUE,
        draw = TRUE, border = par("fg"), col = NA, ...)

```

xy	xy.coords
shape	
open	
repEnds	
draw	
border	border = NA
col	NA
...	ltyxpdlendljoinlmitre

repEnds

draw = TRUENULLxy[lines](#)[polygon](#)

<https://dept-info.labri.fr/~schlick/DOC/sig1.html>

`polygon`

`par`

based on examples in ?grid.xspline

```
xsplineTest <- function(s, open = TRUE,
                        x = c(1,1,3,3)/4,
                        y = c(1,3,3,1)/4, ...) {
  plot(c(0,1), c(0,1), type = "n", axes = FALSE, xlab = "", ylab = "")
  points(x, y, pch = 19)
  xspline(x, y, s, open, ...)
  text(x+0.05*c(-1,-1,1,1), y+0.05*c(-1,1,1,-1), s)
}

op <- par(mfrow = c(3,3), mar = rep(0,4), oma = c(0,0,2,0))
xsplineTest(c(0, -1, -1, 0))
xsplineTest(c(0, -1, 0, 0))
xsplineTest(c(0, -1, 1, 0))
xsplineTest(c(0, 0, -1, 0))
xsplineTest(c(0, 0, 0, 0))
xsplineTest(c(0, 0, 1, 0))
xsplineTest(c(0, 1, -1, 0))
xsplineTest(c(0, 1, 0, 0))
xsplineTest(c(0, 1, 1, 0))
title("Open X-splines", outer = TRUE)

par(mfrow = c(3,3), mar = rep(0,4), oma = c(0,0,2,0))
xsplineTest(c(0, -1, -1, 0), FALSE, col = "grey80")
xsplineTest(c(0, -1, 0, 0), FALSE, col = "grey80")
xsplineTest(c(0, -1, 1, 0), FALSE, col = "grey80")
xsplineTest(c(0, 0, -1, 0), FALSE, col = "grey80")
xsplineTest(c(0, 0, 0, 0), FALSE, col = "grey80")
xsplineTest(c(0, 0, 1, 0), FALSE, col = "grey80")
xsplineTest(c(0, 1, -1, 0), FALSE, col = "grey80")
xsplineTest(c(0, 1, 0, 0), FALSE, col = "grey80")
xsplineTest(c(0, 1, 1, 0), FALSE, col = "grey80")
title("Closed X-splines", outer = TRUE)

par(op)

x <- sort(stats::rnorm(5))
y <- sort(stats::rnorm(5))
plot(x, y, pch = 19)
res <- xspline(x, y, 1, draw = FALSE)
lines(res)
## the end points may be very close together,
## so use last few for direction
nr <- length(res$x)
arrows(res$x[1], res$y[1], res$x[4], res$y[4], code = 1, length = 0.1)
arrows(res$x[nr-3], res$y[nr-3], res$x[nr], res$y[nr], code = 2, length = 0.1)
```


grid

grid-package

grid	grid../doc/grid.pdf
displaylist	grid../doc/displaylist.pdf
frame	../doc/frame.pdf
grobs	grid../doc/grobs.pdf
interactive	grid../doc/interactive.pdf
locndimn	../doc/locndimn.pdf
moveline	../doc/moveline.pdf
nonfinite	grid../doc/nonfinite.pdf
plotexample	grid../doc/plotexample.pdf
rotated	../doc/rotated.pdf
saveload	../doc/saveload.pdf
sharing	../doc/sharing.pdf
viewports	grid../doc/viewports.pdf

```
library(help="grid")
```

<paul@stat.auckland.ac.nz>

<R-core@r-project.org>

`absolute.size`

`"null"`

`absolute.size(unit)`

`unit` `"unit"`

`"inches" "cm" "lines" "npc" "native"`
`widthDetails heightDetails`

`"unit"`

`widthDetails heightDetails`

`arrow`

`grid.lines`

`arrow(angle = 30, length = unit(0.25, "inches"),
 ends = "last", type = "open")`

`angle`

`length`

`ends` `"last" "first" "both"`

`type` `"open" "closed"`

`arrow()
str(arrow(type = "closed"), give.attr=FALSE)`

as.mask

```
as.mask(x, type=c("alpha", "luminance"))
```

x

type

xfigpictex

"GridMask"

viewport

```
## NOTE: on devices without support for masks normal line segments
##       will be drawn
grid.newpage()
## Alpha mask
grid.segments(y0=1, y1=0, gp=gpar(col=2, lwd=100))
pushViewport(viewport(mask=circleGrob(gp=gpar(fill=rgb(0,0,0,.5)))))
grid.segments(gp=gpar(col=3, lwd=100))
grid.newpage()
## Luminance mask
grid.segments(y0=1, y1=0, gp=gpar(col=2, lwd=100))
pushViewport(viewport(mask=as.mask(circleGrob(gp=gpar(fill="grey50")),
                                   "luminance")))
grid.segments(gp=gpar(col=3, lwd=100))
```

`calcStringMetric`

`calcStringMetric(text)``text``stringAscentstringDescentgrobAscentgrobDescent`

```
grid.newpage()
grid.segments(.01, .5, .99, .5, gp=gpar(col="grey"))
metrics <- calcStringMetric(letters)
grid.rect(x=1:26/27,
          width=unit(metrics$width, "inches"),
          height=unit(metrics$ascent, "inches"),
          just="bottom",
          gp=gpar(col="red"))
grid.rect(x=1:26/27,
          width=unit(metrics$width, "inches"),
          height=unit(metrics$descent, "inches"),
          just="top",
          gp=gpar(col="red"))
grid.text(letters, x=1:26/27, just="bottom")

test <- function(x) {
  grid.text(x, just="bottom")
  metric <- calcStringMetric(x)
  if (is.character(x)) {
    grid.rect(width=unit(metric$width, "inches"),
              height=unit(metric$ascent, "inches"),
              just="bottom",
              gp=gpar(col=rgb(1,0,0,.5)))
    grid.rect(width=unit(metric$width, "inches"),
              height=unit(metric$descent, "inches"),
              just="top",
              gp=gpar(col=rgb(1,0,0,.5)))
  }
}
```

```

    } else {
      grid.rect(width=unit(metric$width, "inches"),
                y=unit(.5, "npc") + unit(metric[2], "inches"),
                height=unit(metric$ascent, "inches"),
                just="bottom",
                gp=gpar(col=rgb(1,0,0,.5)))
      grid.rect(width=unit(metric$width, "inches"),
                height=unit(metric$descent, "inches"),
                just="bottom",
                gp=gpar(col=rgb(1,0,0,.5)))
    }
  }

tests <- list("t",
              "test",
              "testy",
              "test\ntwo",
              expression(x),
              expression(y),
              expression(x + y),
              expression(a + b),
              expression(atop(x + y, 2)))

grid.newpage()
nrowcol <- n2mfrow(length(tests))
pushViewport(viewport(layout=grid.layout(nrowcol[1], nrowcol[2]),
                                gp=gpar(cex=5, lwd=.5)))
for (i in 1:length(tests)) {
  col <- (i - 1) %% nrowcol[2] + 1
  row <- (i - 1) %% nrowcol[2] + 1
  pushViewport(viewport(layout.pos.row=row, layout.pos.col=col))
  test(tests[[i]])
  popViewport()
}

```

dataViewport

```

dataViewport(xData = NULL, yData = NULL, xscale = NULL,
             yscale = NULL, extension = 0.05, ...)

```

```

xData
yData
xscale
yscale
extension
...      viewport()

```


[xscalexDataxDataextension](#)

[viewportplotViewport](#)

depth

```
depth(x, ...)
## S3 method for class 'viewport'
depth(x, ...)
## S3 method for class 'path'
depth(x, ...)
```

```
x
...
```

[viewportvpPathgPath](#)

```
vp <- viewport()
depth(vp)
depth(vpStack(vp, vp))
depth(vpList(vpStack(vp, vp), vp))
depth(vpPath("vp"))
depth(vpPath("vp1", "vp2"))
```

deviceLoc

```
deviceLoc(x, y, valueOnly = FALSE, device = FALSE)
deviceDim(w, h, valueOnly = FALSE, device = FALSE)
```

```
xywh
valueOnly      TRUE
device
```

```
convertX()
convertX()
```

```
valueOnlyTRUE
```

unit

```
## A tautology
grid.newpage()
pushViewport(viewport())
deviceLoc(unit(1, "inches"), unit(1, "inches"))

## Something less obvious
grid.newpage()
pushViewport(viewport(width=.5, height=.5))
grid.rect()
x <- unit(1, "in")
y <- unit(1, "in")
grid.circle(x, y, r=unit(2, "mm"))
loc <- deviceLoc(x, y)
loc
upViewport()
grid.circle(loc$x, loc$y, r=unit(1, "mm"), gp=gpar(fill="black"))
```

```

## Something even less obvious
grid.newpage()
pushViewport(viewport(width=.5, height=.5, angle=30))
grid.rect()
x <- unit(.2, "npc")
y <- unit(2, "in")
grid.circle(x, y, r=unit(2, "mm"))
loc <- deviceLoc(x, y)
loc
upViewport()
grid.circle(loc$x, loc$y, r=unit(1, "mm"), gp=gpar(fill="black"))

```

drawDetails

```

drawDetails(x, recording)
preDrawDetails(x)
postDrawDetails(x)

```

```

x
recording

```

[makeContentmakeContext](#)

```

grid.draw
preDrawDetailsvpchildrenvp
drawDetailschildren
postDrawDetailspreDrawDetails
preDrawDetailspostDrawDetails"groewidth""grobheight"

```

[grid.drawmakeContent](#)

`editDetails`

`grid.editeditGrob`

`editDetails(x, specs)`

`x`

`specs`

`grid.editeditGrobgrid::editDetails.xaxis`

[grid.edit](#)

`editViewport`

`editViewport(vp=current.viewport(), ...)`

`vp`

`... viewport()`

[viewport](#)

explode

```
explode(x)
## S3 method for class 'character'
explode(x)
## S3 method for class 'path'
explode(x)
```

x

vpPathgPath

```
explode("vp1::vp2")
explode(vpPath("vp1", "vp2"))
```

gEdit

```
gEditgEditListeditGrob
applyEditapplyEdits
```

```
gEdit(...)
gEditList(...)
applyEdit(x, edit)
applyEdits(x, edits)
```

```
...          editGrobgEdit" gEdit" gEditList
x
edit          "gEdit"
edits         "gEdit""gEditList"
```

```
gEdit" gEdit"  
gEditList" gEditList"  
applyEdit applyEditList
```

[grobeditGrob](#)

```
grid.rect(gp=gpar(col="red"))  
# same thing, but more verbose  
grid.draw(applyEdit(rectGrob(), gEdit(gp=gpar(col="red"))))
```

getNames

getNames()

```
grid.grill()  
getNames()
```

gpar

```
gpar() "gpar"  
get.gpar()
```

```
gpar(...)  
get.gpar(names = NULL)
```

...
names

gp" gpar"" gpar" gp
fill

col
fill
alpha
lty
lwd
lex
lineend
linejoin
linemitre
fontsize
cex fontsize
fontfamily
fontface
lineheight
font fontface

parlendljoinlmitrefamily
rgbcolors
alpha
fill
fontsizecexfontsizecexlineheight
cexcexcexcex
alphalex
fontfamilyfontfamilyHersheySerif
fontface"plain""bold""italic""oblique""bold.italic""cyrillic""cyrillic.oblique"
"EUC"

get.gpar()

"gpar"

Hershey

```

gp <- get.gpar()
utils::str(gp)
## These *do* nothing but produce a "gpar" object:
gpar(col = "red")
gpar(col = "blue", lty = "solid", lwd = 3, fontsize = 16)
get.gpar(c("col", "lty"))
grid.newpage()
vp <- viewport(width = .8, height = .8, gp = gpar(col="blue"))
grid.draw(gTree(children=gList(rectGrob(gp = gpar(col="red")),
                                textGrob(paste("The rect is its own colour (red)",
                                                  "but this text is the colour",
                                                  "set by the gTree (green)",
                                                  sep = "\n"))),
            gp = gpar(col="green"), vp = vp))
grid.text("This text is the colour set by the viewport (blue)",
          y = 1, just = c("center", "bottom"),
          gp = gpar(fontsize=20), vp = vp)
grid.newpage()
## example with multiple values for a parameter
pushViewport(viewport())
grid.points(1:10/11, 1:10/11, gp = gpar(col=1:10))
popViewport()

```

gPath

grid.edit

gPath(...)

...

gPath

[grobedit](#)[Grobadd](#)[Grobremove](#)[Grobget](#)[Grobset](#)[Grob](#)

gPath("g1", "g2")

Grid

```
library(help = grid)
```

[viewportgrid.layoutunit](#)

```
## Diagram of a simple layout
grid.show.layout(grid.layout(4,2,
                             heights=unit(rep(1, 4),
                                           c("lines", "lines", "lines", "null")),
                             widths=unit(c(1, 1), "inches")))
## Diagram of a sample viewport
grid.show.viewport(viewport(x=0.6, y=0.6,
                             width=unit(1, "inches"), height=unit(1, "inches")))
## A flash plotting example
grid.multipanel(vp=viewport(0.5, 0.5, 0.8, 0.8))
```

Grid Viewports

```
viewport(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
         width = unit(1, "npc"), height = unit(1, "npc"),
         default.units = "npc", just = "centre",
         gp = gpar(), clip = "inherit", mask = "inherit",
         xscale = c(0, 1), yscale = c(0, 1),
         angle = 0,
         layout = NULL,
         layout.pos.row = NULL, layout.pos.col = NULL,
         name = NULL)
vpList(...)
vpStack(...)
vpTree(parent, children)
```

```

x
y
width
height
default.units  xywidthheight
just           "left" "right" "centre" "center" "bottom" "top"
gp            "gpar" gpar
clip          "on" "inherit" "off" TRUE "on" FALSE "inherit"
              as.path
mask          "none" FALSE "inherit" TRUE as.mask
xscale
yscale
angle
layout
layout.pos.row
layout.pos.col
name
...
parent
children

```

```

layout.pos.row layout.pos.col NULL locn layout.pos.row layout.pos.col NULL
layout.pos.row = 1 layout.pos.col = NULL NULL layout.pos.row layout.pos.col
layout.pos.row layout.pos.col layout.pos.row = c(1, 3) layout.pos.col = c(2,
4)

```

```

angleclip
clip.as.path
"ROOT" downViewport seekViewport::
vpList vpStack vpTree

```

```
viewport
```

```

pushViewport popViewport downViewport seekViewport upViewport unitgrid.layout
grid.show.layout

```

```

# Diagram of a sample viewport
grid.show.viewport(viewport(x=0.6, y=0.6,
                           width=unit(1, "inches"), height=unit(1, "inches")))

# Demonstrate viewport clipping
clip.demo <- function(i, j, clip1, clip2) {
  pushViewport(viewport(layout.pos.col=i,
                        layout.pos.row=j))
  pushViewport(viewport(width=0.6, height=0.6, clip=clip1))
  grid.rect(gp=gpar(fill="white"))
  grid.circle(r=0.55, gp=gpar(col="red", fill="pink"))
  popViewport()
  pushViewport(viewport(width=0.6, height=0.6, clip=clip2))
  grid.polygon(x=c(0.5, 1.1, 0.6, 1.1, 0.5, -0.1, 0.4, -0.1),
              y=c(0.6, 1.1, 0.5, -0.1, 0.4, -0.1, 0.5, 1.1),
              gp=gpar(col="blue", fill="light blue"))
  popViewport(2)
}

grid.newpage()
grid.rect(gp=gpar(fill="grey"))
pushViewport(viewport(layout=grid.layout(2, 2)))
clip.demo(1, 1, FALSE, FALSE)
clip.demo(1, 2, TRUE, FALSE)
clip.demo(2, 1, FALSE, TRUE)
clip.demo(2, 2, TRUE, TRUE)
popViewport()
# Demonstrate turning clipping off
grid.newpage()
pushViewport(viewport(width=.5, height=.5, clip="on"))
grid.rect()
grid.circle(r=.6, gp=gpar(lwd=10))
pushViewport(viewport(clip="inherit"))
grid.circle(r=.6, gp=gpar(lwd=5, col="grey"))
pushViewport(viewport(clip="off"))
grid.circle(r=.6)
popViewport(3)
# Demonstrate vpList, vpStack, and vpTree
grid.newpage()
tree <- vpTree(viewport(width=0.8, height=0.8, name="A"),
               vpList(vpStack(viewport(x=0.1, y=0.1, width=0.5, height=0.5,
                                   just=c("left", "bottom"), name="B"),
                         viewport(x=0.1, y=0.1, width=0.5, height=0.5,
                                   just=c("left", "bottom"), name="C"),
                         viewport(x=0.1, y=0.1, width=0.5, height=0.5,
                                   just=c("left", "bottom"), name="D")),
                         viewport(x=0.5, width=0.4, height=0.9,
                                   just="left", name="E")))

pushViewport(tree)
for (i in LETTERS[1:5]) {
  seekViewport(i)
  grid.rect()
  grid.text(current.vpTree(FALSE),
            x=unit(1, "mm"), y=unit(1, "npc") - unit(1, "mm"),
            just=c("left", "top"),
            gp=gpar(fontsize=8))
}

```

grid.add

```
grid.add(gPath, child, strict = FALSE, grep = FALSE,  
         global = FALSE, allDevices = FALSE, redraw = TRUE)  
  
addGrob(gTree, child, gPath = NULL, strict = FALSE, grep = FALSE,  
        global = FALSE, warn = TRUE)  
  
setChildren(x, children)
```

```
gTreex  
gPath      grid.addaddGrob  
child  
children  
strict  
grep       gPathgPathc(TRUE, FALSE)gPath  
global     gPath  
warn  
allDevices  
redraw
```

```
addGrob  
grid.addredrawTRUE  
setChildrenaddGrob
```

```
addGrobgrid.addNULL
```

[grobgetGrobremoveGrob](#)

`grid.bezier`

```
grid.bezier(...)
bezierGrob(x = c(0, 0.5, 1, 0.5), y = c(0.5, 1, 0.5, 0),
           id = NULL, id.lengths = NULL,
           default.units = "npc", arrow = NULL,
           name = NULL, gp = gpar(), vp = NULL)
```

x	
y	
id	xyid
id.lengths	xy
default.units	xy
arrow	arrow
name	
gp	"gpar" gpar
vp	
...	bezierGrob

`grid.bezier`

`xy`

[viewportarrow](#)

[grid.xspline](#)

```
x <- c(0.2, 0.2, 0.4, 0.4)
y <- c(0.2, 0.4, 0.4, 0.2)

grid.newpage()
grid.bezier(x, y)
grid.bezier(c(x, x + .4), c(y + .4, y + .4),
            id=rep(1:2, each=4))
grid.segments(.4, .6, .6, .6)
grid.bezier(x, y,
            gp=gpar(lwd=3, fill="black"),
            arrow=arrow(type="closed"),
            vp=viewport(x=.9))
```

`grid.cap`

`grid.cap()`

NULL

`grid.raster`
`dev.capabilities`

```
dev.new(width=0.5, height=0.5)
grid.rect()
grid.text("hi")
cap <- grid.cap()
dev.off()

if(!is.null(cap))
  grid.raster(cap, width=0.5, height=0.5, interpolate=FALSE)
```

`grid.circle`

```
grid.circle(x=0.5, y=0.5, r=0.5, default.units="npc", name=NULL,
            gp=gpar(), draw=TRUE, vp=NULL)
circleGrob(x=0.5, y=0.5, r=0.5, default.units="npc", name=NULL,
            gp=gpar(), vp=NULL)
```

x
y
r
default.units xywidthheight
name
gp "gpar" **gpar**
draw
vp

grid.circle()drawTRUE
"npc""native"grid.circle()

grid.circle()

viewport

grid.clip

```
grid.clip(...)  
clipGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),  
         width = unit(1, "npc"), height = unit(1, "npc"),  
         just = "centre", hjust = NULL, vjust = NULL,  
         default.units = "npc", name = NULL, vp = NULL)
```

```

x
y
width
height
just          "left""right""centre""center""bottom""top"
hjust         just
vjust         just
default.units xywidthheight
name
vp
...           clipGrob

```

```

grid.clip

```

```

clipGrob

```

```

viewport

```

```

# draw across entire viewport, but clipped
grid.clip(x = 0.3, width = 0.1)
grid.lines(gp=gpar(col="green", lwd=5))
# draw across entire viewport, but clipped (in different place)
grid.clip(x = 0.7, width = 0.1)
grid.lines(gp=gpar(col="red", lwd=5))
# Viewport sets new clip region
pushViewport(viewport(width=0.5, height=0.5, clip=TRUE))
grid.lines(gp=gpar(col="grey", lwd=3))
# Return to original viewport; get
# clip region from previous grid.clip()
# (NOT from previous viewport clip region)
popViewport()
grid.lines(gp=gpar(col="black"))

```

grid.convert

```
convertX(x, unitTo, valueOnly = FALSE)
convertY(x, unitTo, valueOnly = FALSE)
convertWidth(x, unitTo, valueOnly = FALSE)
convertHeight(x, unitTo, valueOnly = FALSE)
convertUnit(x, unitTo,
            axisFrom = "x", typeFrom = "location",
            axisTo = axisFrom, typeTo = typeFrom,
            valueOnly = FALSE)
```

```
x
unitTo      unit
axisFrom    "x" "y"
typeFrom    "location" "dimension"
axisTo      axisFrom
typeTo      typeFrom
valueOnly   TRUE
```

```
convertUnit
```

```
"strwidth" "grobwidth"
unit(1, "inches") unit(0.134, "native")
```

```
valueOnly TRUE
```

```
oneinch <- convertUnit(unit(1, "inches"), "native") oneinch
```

```
unit
```

```

## A tautology
convertX(unit(1, "inches"), "inches")
## The physical units
convertX(unit(2.54, "cm"), "inches")
convertX(unit(25.4, "mm"), "inches")
convertX(unit(72.27, "points"), "inches")
convertX(unit(1/12*72.27, "picas"), "inches")
convertX(unit(72, "bigpts"), "inches")
convertX(unit(1157/1238*72.27, "dida"), "inches")
convertX(unit(1/12*1157/1238*72.27, "cicero"), "inches")
convertX(unit(65536*72.27, "scaledpts"), "inches")
convertX(unit(1/2.54, "inches"), "cm")
convertX(unit(1/25.4, "inches"), "mm")
convertX(unit(1/72.27, "inches"), "points")
convertX(unit(1/(1/12*72.27), "inches"), "picas")
convertX(unit(1/72, "inches"), "bigpts")
convertX(unit(1/(1157/1238*72.27), "inches"), "dida")
convertX(unit(1/(1/12*1157/1238*72.27), "inches"), "cicero")
convertX(unit(1/(65536*72.27), "inches"), "scaledpts")

pushViewport(viewport(width=unit(1, "inches"),
                      height=unit(2, "inches"),
                      xscale=c(0, 1),
                      yscale=c(1, 3)))

## Location versus dimension
convertY(unit(2, "native"), "inches")
convertHeight(unit(2, "native"), "inches")
## From "x" to "y" (the conversion is via "inches")
convertUnit(unit(1, "native"), "native",
            axisFrom="x", axisTo="y")
## Convert several values at once
convertX(unit(c(0.5, 2.54), c("npc", "cm")),
         c("inches", "native"))
popViewport()
## Convert a complex unit
convertX(unit(1, "strwidth", "Hello"), "native")

```

grid.copy

grid.copy(grob)

grob

[grid.grob](#)

`grid.curve`

```
grid.curve(...)
curveGrob(x1, y1, x2, y2, default.units = "npc",
          curvature = 1, angle = 90, ncp = 1, shape = 0.5,
          square = TRUE, squareShape = 1,
          inflect = FALSE, arrow = NULL, open = TRUE,
          debug = FALSE,
          name = NULL, gp = gpar(), vp = NULL)
arcCurvature(theta)
```

x1	
y1	
x2	
y2	
default.units	x1y1x2y2
curvature	
angle	
ncp	
shape	grid.xspline
square	ncpangleTRUEFALSE
squareShape	shapesquareTRUE
inflect	
arrow	arrow
open	
debug	
name	
gp	"gpar" gpar
vp	
...	curveGrob
theta	

grid.curve
arcCurvaturecurvatureethetancp

[viewportgrid.xsplinearow](#)

```
curveTest <- function(i, j, ...) {  
  pushViewport(viewport(layout.pos.col=j, layout.pos.row=i))  
  do.call("grid.curve", c(list(x1=.25, y1=.25, x2=.75, y2=.75), list(...)))  
  grid.text(sub("list\\((.*)\\)", "\\1",  
               deparse(substitute(list(...)))),  
            y=unit(1, "npc"))  
  popViewport()  
}  
# grid.newpage()  
pushViewport(plotViewport(c(0, 0, 1, 0),  
                          layout=grid.layout(2, 1, heights=c(2, 1))))  
pushViewport(viewport(layout.pos.row=1,  
                      layout=grid.layout(3, 3, respect=TRUE)))  
  
curveTest(1, 1)  
curveTest(1, 2, inflect=TRUE)  
curveTest(1, 3, angle=135)  
curveTest(2, 1, arrow=arrow())  
curveTest(2, 2, ncp=8)  
curveTest(2, 3, shape=0)  
curveTest(3, 1, curvature=-1)  
curveTest(3, 2, square=FALSE)  
curveTest(3, 3, debug=TRUE)  
popViewport()  
pushViewport(viewport(layout.pos.row=2,  
                      layout=grid.layout(3, 3)))  
  
curveTest(1, 1)  
curveTest(1, 2, inflect=TRUE)  
curveTest(1, 3, angle=135)  
curveTest(2, 1, arrow=arrow())  
curveTest(2, 2, ncp=8)  
curveTest(2, 3, shape=0)  
curveTest(3, 1, curvature=-1)  
curveTest(3, 2, square=FALSE)  
curveTest(3, 3, debug=TRUE)  
popViewport(2)
```

grid.delay

```
delayGrob(expr, list, name=NULL, gp=NULL, vp=NULL)
grid.delay(expr, list, name=NULL, gp=NULL, vp=NULL)
```

```
expr          expressioncall
list          expr
name
gp            "gpar" gpar
vp
```

```
"delayedgrob" grid.delaymakeContent
expr
grid.record() recordGrob() makeContent() drawDetails()
```

```
recordGraphics recordGraphics
```

```
recordGraphics
```

```
grid.delay({
  w <- convertWidth(unit(1, "inches"), "npc")
  rectGrob(width=w)
},
list())
```

```
grid.display.list
```

```
grid.display.list(on=TRUE)
engine.display.list(on=TRUE)
```

```
on
```

grid.DLapply

grid.DLapply(FUN, ...)

FUN

... FUN

NULL

```
grid.newpage()
grid.rect(width=.4, height=.4, x=.25, y=.75, gp=gpar(fill="black"), name="r1")
grid.rect(width=.4, height=.4, x=.5, y=.5, gp=gpar(fill="grey"), name="r2")
grid.rect(width=.4, height=.4, x=.75, y=.25, gp=gpar(fill="white"), name="r3")
grid.DLapply(function(x) { if (is.grob(x)) x$gp <- gpar(); x })
grid.refresh()
```

`grid.draw`

```
grid.draw(x, recording=TRUE)
```

```
x                "grob"  
recording
```

```
vpgpargpchildrenvpgrid.drawchildren  
preDrawDetailsdrawDetailspostDrawDetails
```

`grob`

```
grid.newpage()  
## Create a graphical object, but don't draw it  
l <- linesGrob()  
## Draw it  
grid.draw(l)
```

`grid.edit`

```
grid.edit(gPath, ..., strict = FALSE, grep = FALSE,  
          global = FALSE, allDevices = FALSE, redraw = TRUE)
```

```
grid.gedit(..., grep = TRUE, global = TRUE)
```

```
editGrob(grob, gPath = NULL, ..., strict = FALSE, grep = FALSE,  
         global = FALSE, warn = TRUE)
```

```
grob
...
gPath      grid.editeditGrob
strict
grep       gPathgPathc(TRUE, FALSE)gPath
global     gPath
warn
allDevices
redraw
```

```
editGrob
grid.editredrawTRUE
editDetailsvalidDetails
grid.geditggrid.edit
```

```
editGrobgrid.editNULL
```

[grobgetGrobaddGrobremoveGrob](#)

```
grid.newpage()
grid.xaxis(name = "xa", vp = viewport(width=.5, height=.5))
grid.edit("xa", gp = gpar(col="red"))
# won't work because no ticks (at is NULL)
try(grid.edit(gPath("xa", "ticks"), gp = gpar(col="green")))
grid.edit("xa", at = 1:4/5)
# Now it should work
try(grid.edit(gPath("xa", "ticks"), gp = gpar(col="green")))
```

```
grid.force
```

```

grid.force(x, ...)
## Default S3 method:
grid.force(x, redraw = FALSE, ...)
## S3 method for class 'gPath'
grid.force(x, strict = FALSE, grep = FALSE, global = FALSE,
           redraw = FALSE, ...)
## S3 method for class 'grob'
grid.force(x, draw = FALSE, ...)
forceGrob(x)
grid.revert(x, ...)
## S3 method for class 'gPath'
grid.revert(x, strict = FALSE, grep = FALSE, global = FALSE,
            redraw = FALSE, ...)
## S3 method for class 'grob'
grid.revert(x, draw = FALSE, ...)

```

```

x          xxx
strict     path
grep       path
global     path
draw
redraw
...

```

```

grid.xaxis()atNULL
grid.ls()grid.edit()
grid.force()grid.ls()grid.edit()
forceGrob()grid.force()
grid.revert()grid.force()

```

```

makeContext()makeContent()preDrawDetails()drawDetails()

```

```

grid.newpage()
pushViewport(viewport(width=.5, height=.5))
# Draw xaxis
grid.xaxis(name="xax")
grid.ls()
# Force xaxis
grid.force()
grid.ls()
# Revert xaxis
grid.revert()
grid.ls()
# Draw and force yaxis
grid.force(yaxisGrob(), draw=TRUE)
grid.ls()
# Revert yaxis
grid.revert()
grid.ls()
# Force JUST xaxis
grid.force("xax")
grid.ls()
# Force ALL
grid.force()
grid.ls()
# Revert JUST xaxis
grid.revert("xax")
grid.ls()

```

grid.frame

grid.packgrid.placepackGrobplaceGrobgrid.pack

```

grid.frame(layout=NULL, name=NULL, gp=gpar(), vp=NULL, draw=TRUE)
frameGrob(layout=NULL, name=NULL, gp=gpar(), vp=NULL)

```

layout

name

vp viewport

gp "gpar" **gpar**

draw

grid.frame()drawTRUE

grid.frame()

grid.pack

```
grid.newpage()
grid.frame(name="gf", draw=TRUE)
grid.pack("gf", rectGrob(gp=gpar(fill="grey")), width=unit(1, "null"))
grid.pack("gf", textGrob("hi there"), side="right")
```

grid.function

```
grid.function(...)
functionGrob(f, n = 101, range = "x", units = "native",
             name = NULL, gp=gpar(), vp = NULL)
```

```
grid.abline(intercept, slope, ...)
```

f	xy
n	f
range	"x""y"
units	xy
intercept	
slope	
...	grid.function()
name	
gp	"gpar" gpar
vp	

```
nfxxy
nrangedim"x"xdim"y"range
grid.abline()interceptslope
```

viewport

```
# abline
# NOTE: in ROOT viewport on screen, (0, 0) at top-left
#       and "native" is pixels!
grid.function(function(x) list(x=x, y=0 + 1*x))
# a more "normal" viewport with default normalized "native" coords
grid.newpage()
pushViewport(viewport())
grid.function(function(x) list(x=x, y=0 + 1*x))
# slightly simpler
grid.newpage()
pushViewport(viewport())
grid.abline()
# sine curve
grid.newpage()
pushViewport(viewport(xscale=c(0, 2*pi), yscale=c(-1, 1)))
grid.function(function(x) list(x=x, y=sin(x)))
# constrained sine curve
grid.newpage()
pushViewport(viewport(xscale=c(0, 2*pi), yscale=c(-1, 1)))
grid.function(function(x) list(x=x, y=sin(x)),
              range=0:1)
# inverse sine curve
grid.newpage()
pushViewport(viewport(xscale=c(-1, 1), yscale=c(0, 2*pi)))
grid.function(function(y) list(x=sin(y), y=y),
              range="y")
# parametric function
grid.newpage()
pushViewport(viewport(xscale=c(-1, 1), yscale=c(-1, 1)))
grid.function(function(t) list(x=cos(t), y=sin(t)),
              range=c(0, 9*pi/5))
# physical abline
grid.newpage()
grid.function(function(x) list(x=x, y=0 + 1*x),
              units="in")
```

grid.get

```
grid.get(gPath, strict = FALSE, grep = FALSE, global = FALSE,
         allDevices = FALSE)
```

```
grid.gget(..., grep = TRUE, global = TRUE)
```

```
getGrob(gTree, gPath, strict = FALSE, grep = FALSE, global = FALSE)
```

```

gTree
gPath      grid.getgetGrob
strict
grep      gPathgPathc(TRUE, FALSE)gPath
global    gPath
allDevices
...      grid.get

```

```

grid.ggetggrid.get

```

[grobaddGrobremoveGrob](#)

```

grid.xaxis(name="xa")
grid.get("xa")
grid.get(gPath("xa", "ticks"))

grid.draw(gTree(name="gt", children=gList(xaxisGrob(name="axis"))))
grid.get(gPath("gt", "axis", "ticks"))

```

`grid.glyph`

```

grid.glyph(...)
glyphGrob(glyphInfo,
          x=.5, y=.5, default.units="npc",
          hjust="centre", vjust="centre",
          gp=gpar(), vp=NULL, name=NULL)

```

```

glyphInfo      "RGlyphInfo"glyphInfo
x
y
default.units  xywidthheight
hjustvjust    xycharacter"left"0glyphJust
name
gp            "gpar"gpar
vp
...          glyphGrob()

```

grid.glyph

glyphInfo

grid.grab

```
grid.grab(warn = 2, wrap = wrap.grobs, wrap.grobs = FALSE, ...)
grid.grabExpr(expr, warn = 2, wrap = wrap.grobs, wrap.grobs = FALSE,
              width = 7, height = 7, device = offscreen, ...)
```

```
expr
warn
wrap          TRUE
wrap.grobs    wrap=TRUE
widthheight
device        pdf
...
```

grid.grabgrid.grabExpr

wrap=TRUEwrap.grobsTRUE

wrap.grobs=TRUEgrobXgrobWidth

gTree


```
# Searching for grobs
grid.grep("parent", sampleGTree)
grid.grep("parent", sampleGTree, strict=TRUE)
grid.grep("grandparent", sampleGTree, strict=TRUE)
grid.grep("grandparent::parent", sampleGTree)
grid.grep("parent::child", sampleGTree)
grid.grep("[a-z]", sampleGTree, grep=TRUE)
grid.grep("[a-z]", sampleGTree, grep=TRUE, global=TRUE)
# Searching for viewports
grid.grep("vp1", sampleGTree, viewports=TRUE)
grid.grep("vp2", sampleGTree, viewports=TRUE)
grid.grep("vp", sampleGTree, viewports=TRUE, grep=TRUE)
grid.grep("vp2", sampleGTree, viewports=TRUE, strict=TRUE)
grid.grep("vp1:vp2", sampleGTree, viewports=TRUE)
# Searching for both
grid.grep("[a-z]", sampleGTree, viewports=TRUE, grep=TRUE, global=TRUE)
```

grid.grill

```
grid.grill(h = unit(seq(0.25, 0.75, 0.25), "npc"),
           v = unit(seq(0.25, 0.75, 0.25), "npc"),
           default.units = "npc", gp=gpar(col = "grey"), vp = NULL)
```

```
h
v
default.units  hv
gp              "gpar" gpar
vp
```

viewport

grid.grob

`grob()gTree()grobTree()gList()`

Grob Creation:

```
grob (... , name = NULL, gp = NULL, vp = NULL, cl = NULL)
gTree(... , name = NULL, gp = NULL, vp = NULL, children = NULL,
       childrenvp = NULL, cl = NULL)
grobTree(... , name = NULL, gp = NULL, vp = NULL,
         childrenvp = NULL, cl = NULL)
gList(...)
```

Grob Properties:

```
childNames(gTree)
is.grob(x)
```

...	grob	gTree	gList	grobTree
name				
children		"gList"		
childrenvp		viewport	NULL	
gp		"gpar"	gpar	
vp		viewport	NULL	
cl				
gTree		"gTree"		
x				

```
"grob""gTree""gList"
grob()gTree()validDetails
"gTree"childrenvpupViewportchildrenvpvpPath
gPath::
childNames
grid.linesgrid.rectgrid.xaxisgrid.yaxis
grobTreegTree
grid.grob
```

"grob"

[grid.drawgrid.editgrid.get](#)

grid.group

```
groupGrob(src, op = "over", dst = NULL, coords = TRUE,
          name = NULL, gp=gpar(), vp=NULL)
grid.group(src, op = "over", dst = NULL, coords = TRUE,
          name = NULL, gp=gpar(), vp=NULL)
```

```
defineGrob(src, op = "over", dst = NULL, coords = TRUE,
           name = NULL, gp=gpar(), vp=NULL)
grid.define(src, op = "over", dst = NULL, coords = TRUE,
           name = NULL, gp=gpar(), vp=NULL)
```

```
useGrob(group, transform=viewportTransform,
        name=NULL, gp=gpar(), vp=NULL)
```

```
grid.use(group, transform=viewportTransform,
        name=NULL, gp=gpar(), vp=NULL)
```

```
src
op
dst
coords
group
transform      viewportTransform
name
gp              "gpar" gpar
vp
```

```
grid.group()
grid.group()srcdst"over""dest.out"dstsrc"clear""source""over""in""out""atop"
"dest""dest.over""dest.in""dest.out""dest.atop""xor""add""saturate""multiply"
"screen""overlay""darken""lighten""color.dodge""color.burn""hard.light"
"soft.light""difference""exclusion"dev.capabilities
grid.define()grid.use()
```

[viewportTransform](#)

[xfigpictexpdf\(\)](#)

```

## NOTE: on devices without support for groups (or masks or patterns),
##       there will only be two overlapping opaque circles
grid.newpage()
pat <- pattern(rasterGrob(matrix(c(.5, 1, 1, .5), nrow=2),
                                width=unit(1, "cm"),
                                height=unit(1, "cm"),
                                interpolate=FALSE,
                                width=unit(1, "cm"), height=unit(1, "cm"),
                                extend="repeat"))
grid.rect(gp=gpar(col=NA, fill=pat))
masks <- dev.capabilities()$masks
if (is.character(masks) && "luminance" %in% masks) {
  mask <- as.mask(rectGrob(gp=gpar(col=NA, fill="grey50")), type="luminance")
} else {
  mask <- rectGrob(gp=gpar(col=NA, fill=rgb(0,0,0,.5)))
}
pushViewport(viewport(mask=mask))
pushViewport(viewport(y=.5, height=.5, just="bottom"))
grid.circle(1:2/3, r=.45, gp=gpar(fill=2:3))
popViewport()
pushViewport(viewport(y=0, height=.5, just="bottom"))
grid.group(circleGrob(1:2/3, r=.45, gp=gpar(fill=2:3)))
popViewport()

```

grid.layout

```

grid.layout(nrow = 1, ncol = 1,
            widths = unit(rep_len(1, ncol), "null"),
            heights = unit(rep_len(1, nrow), "null"),
            default.units = "null", respect = FALSE,
            just="centre")

nrow
ncol
widths
heights
default.units  widthsheights
respect
just           "left""right""centre""center""bottom""top""left"

```

widthsheightsunits"null"

layoutlayoutlayoutgrid.layoutviewport

[grid.show.layoutviewportlayout](#)

```
## A variety of layouts (some a bit mid-bending ...)
layout.torture()
## Demonstration of layout justification
grid.newpage()
testlay <- function(just="centre") {
  pushViewport(viewport(layout=grid.layout(1, 1, widths=unit(1, "inches"),
    heights=unit(0.25, "npc"),
    just=just)))
  pushViewport(viewport(layout.pos.col=1, layout.pos.row=1))
  grid.rect()
  grid.text(paste(just, collapse="-"))
  popViewport(2)
}
testlay()
testlay(c("left", "top"))
testlay(c("right", "top"))
testlay(c("right", "bottom"))
testlay(c("left", "bottom"))
testlay(c("left"))
testlay(c("right"))
testlay(c("bottom"))
testlay(c("top"))
```

grid.lines

```

grid.lines(x = unit(c(0, 1), "npc"),
           y = unit(c(0, 1), "npc"),
           default.units = "npc",
           arrow = NULL, name = NULL,
           gp=gpar(), draw = TRUE, vp = NULL)
linesGrob(x = unit(c(0, 1), "npc"),
          y = unit(c(0, 1), "npc"),
          default.units = "npc",
          arrow = NULL, name = NULL,
          gp=gpar(), vp = NULL)
grid.polyline(...)
polylineGrob(x = unit(c(0, 1), "npc"),
             y = unit(c(0, 1), "npc"),
             id=NULL, id.lengths=NULL,
             default.units = "npc",
             arrow = NULL, name = NULL,
             gp=gpar(), vp = NULL)

```

```

x
y
default.units  xy
arrow          arrow
name
gp             "gpar"gpar
draw
vp
id             xyid
id.lengths    xy
...           polylineGrob

```

```

grid.linesdrawTRUE

```

```

grid.lines

```

[viewportarrow](#)

```

grid.lines()
# Using id (NOTE: locations are not in consecutive blocks)
grid.newpage()
grid.polyline(x=c((0:4)/10, rep(.5, 5), (10:6)/10, rep(.5, 5)),
              y=c(rep(.5, 5), (10:6/10), rep(.5, 5), (0:4)/10),
              id=rep(1:5, 4),
              gp=gpar(col=1:5, lwd=3))
# Using id.lengths
grid.newpage()
grid.polyline(x=outer(c(0, .5, 1, .5), 5:1/5),
              y=outer(c(.5, 1, .5, 0), 5:1/5),
              id.lengths=rep(4, 5),
              gp=gpar(col=1:5, lwd=3))

```

grid.locator

```
grid.locator(unit = "native")
```

```
unit          unit
```

```
locator
```

```
NULL
```

[viewportunitlocatortrellis.focuspanel.identify](#)

```

if (dev.interactive()) {
  ## Need to write a more sophisticated unit as.character method
  unittrim <- function(unit) {
    sub("^[0-9]+|[0-9]+.[0-9][0-9]*", "\\1", as.character(unit))
  }
  do.click <- function(unit) {
    click.locn <- grid.locator(unit)
    grid.segments(unit.c(click.locn$x, unit(0, "npc")),
                  unit.c(unit(0, "npc"), click.locn$y),
                  click.locn$x, click.locn$y,

```

```

        gp=gpar(lty="dashed", col="grey"))
grid.points(click.locn$x, click.locn$y, pch=16, size=unit(1, "mm"))
clickx <- unittrim(click.locn$x)
clicky <- unittrim(click.locn$y)
grid.text(paste0("(", clickx, ", ", clicky, ")"),
          click.locn$x + unit(2, "mm"), click.locn$y,
          just="left")
}

grid.newpage() # (empty slate)
## device
do.click("inches")
Sys.sleep(1)

pushViewport(viewport(width=0.5, height=0.5,
                      xscale=c(0, 100), yscale=c(0, 10)))
grid.rect()
grid.xaxis()
grid.yaxis()
do.click("native")
popViewport()
}

```

grid.ls

```

grid.ls(x=NULL, grobs=TRUE, viewports=FALSE, fullNames=FALSE,
        recursive=TRUE, print=TRUE, flatten=TRUE, ...)

```

```

nestedListing(x, gindent="  ", vpindent=gindent)
pathListing(x, gvpSep=" | ", gAlign=TRUE)
grobPathListing(x, ...)

```

```

x          NULLNULL
          grid.ls

```

```

grobs
viewports
fullNames
recursive
print
flatten
gindent
vpindent
gvpSep
gAlign
...      print

```

xNULL

x

printnestedListingpathListinggrobPathListing

"gridFlatListing"flattenTRUE"gridListing"

[grobviewport](#)

```
# A gTree, called "parent", with childrenvp vpTree (vp2 within vp1)
# and child grob, called "child", with vp vpPath (down to vp2)
sampleGTree <- gTree(name="parent",
                     children=gList(grob(name="child", vp="vp1::vp2")),
                     childrenvp=vpTree(parent=viewport(name="vp1"),
                                       children=vpList(viewport(name="vp2"))))

grid.ls(sampleGTree)
# Show viewports too
grid.ls(sampleGTree, viewports=TRUE)
# Only show viewports
grid.ls(sampleGTree, viewports=TRUE, grobs=FALSE)
# Alternate displays
# nested listing, custom indent
grid.ls(sampleGTree, viewports=TRUE, print=nestedListing, gindent="--")
# path listing
grid.ls(sampleGTree, viewports=TRUE, print=pathListing)
# path listing, without grobs aligned
grid.ls(sampleGTree, viewports=TRUE, print=pathListing, gAlign=FALSE)
# grob path listing
grid.ls(sampleGTree, viewports=TRUE, print=grobPathListing)
# path listing, grobs only
grid.ls(sampleGTree, print=pathListing)
# path listing, viewports only
grid.ls(sampleGTree, viewports=TRUE, grobs=FALSE, print=pathListing)
# raw flat listing
str(grid.ls(sampleGTree, viewports=TRUE, print=FALSE))
```

grid.move.to

```
grid.move.to(x = 0, y = 0, default.units = "npc", name = NULL,  
            draw = TRUE, vp = NULL)
```

```
moveToGrob(x = 0, y = 0, default.units = "npc", name = NULL,  
          vp = NULL)
```

```
grid.line.to(x = 1, y = 1, default.units = "npc",  
            arrow = NULL, name = NULL,  
            gp = gpar(), draw = TRUE, vp = NULL)
```

```
lineToGrob(x = 1, y = 1, default.units = "npc", arrow = NULL,  
          name = NULL, gp = gpar(), vp = NULL)
```

```
x  
y  
default.units  xy  
arrow          arrow  
name  
draw  
gp             "gpar" gpar  
vp
```

```
grid.move.to/line.to()drawTRUE
```

```
grid.move.to/line.to()
```

viewportarrow

```
grid.newpage()  
grid.move.to(0.5, 0.5)  
grid.line.to(1, 1)  
grid.line.to(0.5, 0)  
pushViewport(viewport(x=0, y=0, width=0.25, height=0.25, just=c("left", "bottom")))  
grid.rect()  
grid.grill()  
grid.line.to(0.5, 0.5)  
popViewport()
```

`grid.newpage`

```
grid.newpage(recording = TRUE, clearGroups = TRUE)
```

```
recording
```

```
clearGroups    grid.group
```

```
gpar("fill")X11windowsquartz
```

```
"before.grid.newpage""grid.newpage"setHookget
```

`grid.null`

```
nullGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),  
          default.units = "npc",  
          name = NULL, vp = NULL)  
grid.null(...)
```

```
x
```

```
y
```

```
default.units  xywidthheight
```

```
name
```

```
vp
```

```
...           nullGrob()
```

viewport

```
grid.newpage()
grid.null(name="ref")
grid.rect(height=grobHeight("ref"))
grid.segments(0, 0, grobX("ref", 0), grobY("ref", 0))
```

grid.pack

grid.frameframeGrobgrid.frameframeGrob

```
grid.pack(gPath, grob, redraw = TRUE, side = NULL,
          row = NULL, row.before = NULL, row.after = NULL,
          col = NULL, col.before = NULL, col.after = NULL,
          width = NULL, height = NULL,
          force.width = FALSE, force.height = FALSE, border = NULL,
          dynamic = FALSE)
```

```
packGrob(frame, grob, side = NULL,
          row = NULL, row.before = NULL, row.after = NULL,
          col = NULL, col.before = NULL, col.after = NULL,
          width = NULL, height = NULL,
          force.width = FALSE, force.height = FALSE, border = NULL,
          dynamic = FALSE)
```

gPath	
frame	framegrid.frame
grob	grob
redraw	
side	"left""top""right""bottom"
row	NULL
row.before	
row.after	
col	NULL
col.before	

```
col.after
width
height
force.width    grid.pack
force.height   grid.pack
border         unit
dynamic        "grobwidth""grobheight"
```

```
packGrob
grid.packredrawTRUE
```

```
widthheight
```

```
packGrobgrid.packNULL
```

```
grid.framegrid.placegrid.editgPath
```

```
grid.path
```

```
pathGrob(x, y,
          id=NULL, id.lengths=NULL,
          pathId=NULL, pathId.lengths=NULL,
          rule="winding",
          default.units="npc",
          name=NULL, gp=gpar(), vp=NULL)
grid.path(...)
```

```
x
y
id          xyid
id.lengths  xy
pathId       xypathId
pathId.lengths xy
rule         "winding""evenodd"
default.units xy
name
gp           "gpar"gpar
vp
...          pathGrob()
```

```
grid.pathdrawTRUE
```

```
xfigpictex
```

```
viewport
```

```
pathSample <- function(x, y, rule, gp = gpar()) {  
  if (is.na(rule))  
    grid.path(x, y, id = rep(1:2, each = 4), gp = gp)  
  else  
    grid.path(x, y, id = rep(1:2, each = 4), rule = rule, gp = gp)  
  if (!is.na(rule))  
    grid.text(paste("Rule:", rule), y = 0, just = "bottom")  
}
```

```
pathTriplet <- function(x, y, title) {  
  pushViewport(viewport(height = 0.9, layout = grid.layout(1, 3),  
    gp = gpar(cex = .7)))  
  grid.rect(y = 1, height = unit(1, "char"), just = "top",  
    gp = gpar(col = NA, fill = "grey"))  
  grid.text(title, y = 1, just = "top")  
  pushViewport(viewport(layout.pos.col = 1))  
  pathSample(x, y, rule = "winding",  
    gp = gpar(fill = "grey"))  
  popViewport()  
  pushViewport(viewport(layout.pos.col = 2))  
  pathSample(x, y, rule = "evenodd",  
    gp = gpar(fill = "grey"))  
  popViewport()  
  pushViewport(viewport(layout.pos.col = 3))  
  pathSample(x, y, rule = NA)  
  popViewport()  
  popViewport()  
}
```

```
pathTest <- function() {  
  grid.newpage()  
  pushViewport(viewport(layout = grid.layout(5, 1)))  
  pushViewport(viewport(layout.pos.row = 1))  
  pathTriplet(c(.1, .1, .9, .9, .2, .2, .8, .8),  
    c(.1, .9, .9, .1, .2, .8, .8, .2),  
    "Nested rectangles, both clockwise")  
  popViewport()  
  pushViewport(viewport(layout.pos.row = 2))
```

```

    pathTriplet(c(.1, .1, .9, .9, .2, .8, .8, .2),
                c(.1, .9, .9, .1, .2, .2, .8, .8),
                "Nested rectangles, outer clockwise, inner anti-clockwise")
    popViewport()
    pushViewport(viewport(layout.pos.row = 3))
    pathTriplet(c(.1, .1, .4, .4, .6, .9, .9, .6),
                c(.1, .4, .4, .1, .6, .6, .9, .9),
                "Disjoint rectangles")
    popViewport()
    pushViewport(viewport(layout.pos.row = 4))
    pathTriplet(c(.1, .1, .6, .6, .4, .4, .9, .9),
                c(.1, .6, .6, .1, .4, .9, .9, .4),
                "Overlapping rectangles, both clockwise")
    popViewport()
    pushViewport(viewport(layout.pos.row = 5))
    pathTriplet(c(.1, .1, .6, .6, .4, .9, .9, .4),
                c(.1, .6, .6, .1, .4, .4, .9, .9),
                "Overlapping rectangles, one clockwise, other anti-clockwise")
    popViewport()
    popViewport()
}

pathTest()

# Drawing multiple paths at once
holed_rect <- cbind(c(.15, .15, -.15, -.15, .1, .1, -.1, -.1),
                  c(.15, -.15, -.15, .15, .1, -.1, -.1, .1))
holed_rects <- rbind(
  holed_rect + matrix(c(.7, .2), nrow = 8, ncol = 2, byrow = TRUE),
  holed_rect + matrix(c(.7, .8), nrow = 8, ncol = 2, byrow = TRUE),
  holed_rect + matrix(c(.2, .5), nrow = 8, ncol = 2, byrow = TRUE)
)
grid.newpage()
grid.path(x = holed_rects[, 1], y = holed_rects[, 2],
          id = rep(1:6, each = 4), pathId = rep(1:3, each = 8),
          gp = gpar(fill = c('red', 'blue', 'green')),
          rule = 'evenodd')

# Not specifying pathId will treat all points as part of the same path, thus
# having same fill
grid.newpage()
grid.path(x = holed_rects[, 1], y = holed_rects[, 2],
          id = rep(1:6, each = 4),
          gp = gpar(fill = c('red', 'blue', 'green')),
          rule = 'evenodd')

```

```
grid.place
```

```
grid.pack()packGrob
```

```

grid.place(gPath, grob, row = 1, col = 1, redraw = TRUE)
placeGrob(frame, grob, row = NULL, col = NULL)

```

gPath
frame framegrid.frame
grob grob
row
col
redraw

placeGrob
grid.placeredrawTRUE

placeGrobgrid.placeNULL

[grid.framegrid.packgrid.editgPath](#)

grid.plot.and.legend

grid.plot.and.legend()

grid.plot.and.legend()

grid.points

```
grid.points(x = stats::runif(10),
            y = stats::runif(10),
            pch = 1, size = unit(1, "char"),
            default.units = "native", name = NULL,
            gp = gpar(), draw = TRUE, vp = NULL)
pointsGrob(x = stats::runif(10),
           y = stats::runif(10),
           pch = 1, size = unit(1, "char"),
           default.units = "native", name = NULL,
           gp = gpar(), vp = NULL)
```

```
x
y
pch           pointsfill
size
default.units xy
name
gp            "gpar" gparfillbgpoints pch = 21:25
draw
vp
```

```
grid.pointsdrawTRUE
```

```
grobgrid.points
```

```
viewport
```

grid.polygon

```
grid.polygon(x=c(0, 0.5, 1, 0.5), y=c(0.5, 1, 0.5, 0),
             id=NULL, id.lengths=NULL,
             default.units="npc", name=NULL,
             gp=gpar(), draw=TRUE, vp=NULL)
polygonGrob(x=c(0, 0.5, 1, 0.5), y=c(0.5, 1, 0.5, 0),
            id=NULL, id.lengths=NULL,
            default.units="npc", name=NULL,
            gp=gpar(), vp=NULL)
```

```
x
y
id          xyid
id.lengths  xy
default.units xywidthheight
name
gp          "gpar" gpar
draw
vp
```

```
grid.polygondrawTRUE
```

viewport

```
grid.polygon()
# Using id (NOTE: locations are not in consecutive blocks)
grid.newpage()
grid.polygon(x=c((0:4)/10, rep(.5, 5), (10:6)/10, rep(.5, 5)),
             y=c(rep(.5, 5), (10:6/10), rep(.5, 5), (0:4)/10),
             id=rep(1:5, 4),
             gp=gpar(fill=1:5))
# Using id.lengths
grid.newpage()
grid.polygon(x=outer(c(0, .5, 1, .5), 5:1/5),
             y=outer(c(.5, 1, .5, 0), 5:1/5),
             id.lengths=rep(4, 5),
             gp=gpar(fill=1:5))
```

grid.pretty

n
GEpretty()[axis\(\)](#)[axTicks\(\)](#)

grid.pretty(range, n = 5)

range [range\(\)](#)
n

grid.raster

```
grid.raster(image,  
             x = unit(0.5, "npc"), y = unit(0.5, "npc"),  
             width = NULL, height = NULL,  
             just = "centre", hjust = NULL, vjust = NULL,  
             interpolate = TRUE, default.units = "npc",  
             name = NULL, gp = gpar(), vp = NULL)
```

```
rasterGrob(image,  
            x = unit(0.5, "npc"), y = unit(0.5, "npc"),  
            width = NULL, height = NULL,  
            just = "centre", hjust = NULL, vjust = NULL,  
            interpolate = TRUE, default.units = "npc",  
            name = NULL, gp = gpar(), vp = NULL)
```

image
x
y
width
height

```
just          "left""right""centre""center""bottom""top"
hjust         just
vjust         just
default.units xywidthheight
name
gp            "gpar" gpar
vp
interpolate
```

```
widthheightwidthheight
```

```
rasterImage
```

```
gpalp
```

```
as.raster
```

```
dev.capabilities
```

```
redGradient <- matrix(hcl(0, 80, seq(50, 80, 10)),
                      nrow=4, ncol=5)

# interpolated
grid.newpage()
grid.raster(redGradient)
# blocky
grid.newpage()
grid.raster(redGradient, interpolate=FALSE)
# blocky and stretched
grid.newpage()
grid.raster(redGradient, interpolate=FALSE, height=unit(1, "npc"))

# The same raster drawn several times
grid.newpage()
grid.raster(0, x=1:3/4, y=1:3/4, width=.1, interpolate=FALSE)
```

grid.record

```
recordGrob(expr, list, name=NULL, gp=NULL, vp=NULL)
grid.record(expr, list, name=NULL, gp=NULL, vp=NULL)
```

expr	expressioncall
list	expr
name	
gp	"gpar" gpar
vp	

"recordedGrob"grid.recorddrawDetails

recordGraphicsrecordGraphics

[recordGraphics](#)

```
grid.record({
  w <- convertWidth(unit(1, "inches"), "npc")
  grid.rect(width=w)
},
list())
```

grid.rect

```
grid.rect(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
          width = unit(1, "npc"), height = unit(1, "npc"),
          just = "centre", hjust = NULL, vjust = NULL,
          default.units = "npc", name = NULL,
          gp=gpar(), draw = TRUE, vp = NULL)
rectGrob(x = unit(0.5, "npc"), y = unit(0.5, "npc"),
          width = unit(1, "npc"), height = unit(1, "npc"),
          just = "centre", hjust = NULL, vjust = NULL,
          default.units = "npc", name = NULL,
          gp=gpar(), vp = NULL)
```

x

y

width

height

just "left""right""centre""center""bottom""top"

hjust just

vjust just

default.units xywidthheight

name

gp "gpar"[gpar](#)

draw

vp

grid.rectdrawTRUE

grid.rect

[viewport](#)

`grid.refresh`

`grid.refresh()`

`grid.remove`

```
grid.remove(gPath, warn = TRUE, strict = FALSE, grep = FALSE,
            global = FALSE, allDevices = FALSE, redraw = TRUE)
```

```
grid.gremove(..., grep = TRUE, global = TRUE)
```

```
removeGrob(gTree, gPath, strict = FALSE, grep = FALSE,
            global = FALSE, warn = TRUE)
```

`gTree`

`gPath` `grid.remove``removeGrob``gTree`

`strict` [logical](#)`gPath`

`grep` [logical](#)`gPath``gPath``hc(TRUE, FALSE)``gPath`

`global` [logical](#)`gPath`

`allDevices` [logical](#)

`warn`

`redraw`

`...` [grid.get](#)

`removeGrob`

`grid.remove``redraw``TRUE`

`grid.gremove``grid.remove`

`removeGrob``grid.remove``NULL`

[grob](#)`getGrob``addGrob`

`grid.reorder`

```
grid.reorder(gPath, order, back=TRUE, grep=FALSE, redraw=TRUE)
reorderGrob(x, order, back=TRUE)
```

```
gPath
```

```
x
```

```
order
```

```
back          order
```

```
grep          gPath
```

```
redraw
```

```
order
```

```
orderorderback=FALSEorderorder
```

```
grid.grab()
```

```
grid.reorder()reorderGrob()
```

```
grid.reorder()reorderGrob()
```

```
# gTree with two children, "red-rect" and "blue-rect" (in that order)
gt <- gTree(children=gList(
  rectGrob(gp=gpar(col=NA, fill="red"),
    width=.8, height=.2, name="red-rect"),
  rectGrob(gp=gpar(col=NA, fill="blue"),
    width=.2, height=.8, name="blue-rect")),
  name="gt")
grid.newpage()
grid.draw(gt)
# Spec entire order as numeric (blue-rect, red-rect)
grid.reorder("gt", 2:1)
# Spec entire order as character
grid.reorder("gt", c("red-rect", "blue-rect"))
# Only spec the one I want behind as character
grid.reorder("gt", "blue-rect")
# Only spec the one I want in front as character
grid.reorder("gt", "blue-rect", back=FALSE)
```

grid.segments

```
grid.segments(x0 = unit(0, "npc"), y0 = unit(0, "npc"),
              x1 = unit(1, "npc"), y1 = unit(1, "npc"),
              default.units = "npc",
              arrow = NULL,
              name = NULL, gp = gpar(), draw = TRUE, vp = NULL)
segmentsGrob(x0 = unit(0, "npc"), y0 = unit(0, "npc"),
             x1 = unit(1, "npc"), y1 = unit(1, "npc"),
             default.units = "npc",
             arrow = NULL, name = NULL, gp = gpar(), vp = NULL)
```

x0

y0

x1

y1

default.units

arrow arrow

name

gp "gpar"[gpar](#)

draw

vp

grid.segmentsdrawTRUE

grid.segments

[viewportarrow](#)

`grid.set`

```
grid.set(gPath, newGrob, strict = FALSE, grep = FALSE,  
         redraw = TRUE)
```

```
setGrob(gTree, gPath, newGrob, strict = FALSE, grep = FALSE)
```

```
gTree  
gPath      grid.setsetGrob  
newGrob  
strict  
grep       gPathgPathc(TRUE, FALSE)gPath  
redraw
```

```
setGrob  
grid.setredrawTRUE
```

```
setGrobgrid.setNULL
```

[grid.grob](#)

`grid.show.layout`

```
grid.show.layout(1, newpage=TRUE, vp.ex = 0.8, bg = "light grey",  
                 cell.border = "blue", cell.fill = "light blue",  
                 cell.label = TRUE, label.col = "blue",  
                 unit.col = "red", vp = NULL, ...)
```

```

l
newpage
vp.ex      (0,1]
bg
cell.border
cell.fill
cell.label
label.col
unit.col
vp
...        format

vp

```

viewportgrid.layout

```

## Diagram of a simple layout
grid.show.layout(grid.layout(4,2,
    heights=unit(rep(1, 4),
        c("lines", "lines", "lines", "null")),
    widths=unit(c(1, 1), "inches")))

```

```
grid.show.viewport
```

```

grid.show.viewport(v, parent.layout = NULL, newpage = TRUE,
    vp.ex = 0.8, border.fill="light grey",
    vp.col="blue", vp.fill="light blue",
    scale.col="red",
    vp = NULL)

```

```
v
parent.layout vv
newpage
vp.ex (0,1]
border.fill
vp.col
vp.fill
scale.col
vp
```

```
vp
```

viewport

```
## Diagram of a sample viewport
grid.show.viewport(viewport(x=0.6, y=0.6,
                             width=unit(1, "inches"), height=unit(1, "inches")))
grid.show.viewport(viewport(layout.pos.row=2, layout.pos.col=2:3),
                    grid.layout(3, 4))
```

```
grid.stroke
```

```
strokeGrob(x, ...)
## S3 method for class 'grob'
strokeGrob(x, name=NULL, gp=gpar(), vp=NULL, ...)
## S3 method for class 'GridPath'
strokeGrob(x, name=NULL, vp=NULL, ...)
grid.stroke(...)
fillGrob(x, ...)
## S3 method for class 'grob'
fillGrob(x, rule=c("winding", "evenodd"),
          name=NULL, gp=gpar(), vp=NULL, ...)
## S3 method for class 'GridPath'
```

```

fillGrob(x, name=NULL, vp=NULL, ...)
grid.fill(...)
fillStrokeGrob(x, ...)
## S3 method for class 'grob'
fillStrokeGrob(x, rule=c("winding", "evenodd"),
               name=NULL, gp=gpar(), vp=NULL, ...)
## S3 method for class 'GridPath'
fillStrokeGrob(x, name=NULL, vp=NULL, ...)
grid.fillStroke(...)
as.path(x, gp=gpar(), rule=c("winding", "evenodd"))

```

```

x          as.path()
rule
name
gp         "gpar" gpar
vp
...        grid.*()*Grob()

```

```

x
grid.stroke()
grid.fill()

```

```

as.path()viewport
xfigpictex

```

```

## NOTE: on devices without support for stroking and filling
##       nothing will be drawn
grid.newpage()
grid.stroke(textGrob("hello", gp=gpar(cex=10)))
grid.fill(circleGrob(1:2/3, r=.3), gp=gpar(fill=rgb(1,0,0,.5)))

```

grid.text

```
grid.text(label, x = unit(0.5, "npc"), y = unit(0.5, "npc"),
          just = "centre", hjust = NULL, vjust = NULL, rot = 0,
          check.overlap = FALSE, default.units = "npc",
          name = NULL, gp = gpar(), draw = TRUE, vp = NULL)

textGrob(label, x = unit(0.5, "npc"), y = unit(0.5, "npc"),
          just = "centre", hjust = NULL, vjust = NULL, rot = 0,
          check.overlap = FALSE, default.units = "npc",
          name = NULL, gp = gpar(), vp = NULL)
```

label	as.graphicsAnnot
x	
y	
just	"left""right""centre""center""bottom""top"
hjust	just
vjust	just
rot	
check.overlap	
default.units	xy
name	
gp	"gpar" gpar
draw	
vp	

grid.textdrawTRUE

label

grid.text()

[viewport](#)

```

grid.newpage()
x <- stats::runif(20)
y <- stats::runif(20)
rot <- stats::runif(20, 0, 360)
grid.text("SOMETHING NICE AND BIG", x=x, y=y, rot=rot,
          gp=gpar(fontsize=20, col="grey"))
grid.text("SOMETHING NICE AND BIG", x=x, y=y, rot=rot,
          gp=gpar(fontsize=20), check.overlap=TRUE)

grid.newpage() ## plotmath example
grid.text(quote(frac(e^{-x^2/2}, sqrt(2*pi))), x=x, y=y, rot=stats::runif(20, -45,45),
          gp=gpar(fontsize=17, col=4), check.overlap=TRUE)

grid.newpage()
draw.text <- function(just, i, j) {
  grid.text("ABCD", x=x[j], y=y[i], just=just)
  grid.text(deparse(substitute(just)), x=x[j], y=y[i] + unit(2, "lines"),
            gp=gpar(col="grey", fontsize=8))
}
x <- unit(1:4/5, "npc")
y <- unit(1:4/5, "npc")
grid.grill(h=y, v=x, gp=gpar(col="grey"))
draw.text(c("bottom"), 1, 1)
draw.text(c("left", "bottom"), 2, 1)
draw.text(c("right", "bottom"), 3, 1)
draw.text(c("centre", "bottom"), 4, 1)
draw.text(c("centre"), 1, 2)
draw.text(c("left", "centre"), 2, 2)
draw.text(c("right", "centre"), 3, 2)
draw.text(c("centre", "centre"), 4, 2)
draw.text(c("top"), 1, 3)
draw.text(c("left", "top"), 2, 3)
draw.text(c("right", "top"), 3, 3)
draw.text(c("centre", "top"), 4, 3)
draw.text(c(), 1, 4)
draw.text(c("left"), 2, 4)
draw.text(c("right"), 3, 4)
draw.text(c("centre"), 4, 4)

```

grid.xaxis

```

grid.xaxis(at = NULL, label = TRUE, main = TRUE,
           edits = NULL, name = NULL,
           gp = gpar(), draw = TRUE, vp = NULL)

xaxisGrob(at = NULL, label = TRUE, main = TRUE,
          edits = NULL, name = NULL,
          gp = gpar(), vp = NULL)

```

at	
label	at
main	TRUEFALSE
edits	atNULL
name	
gp	"gpar" gpar
draw	
vp	NULL

grid.xaxisdrawTRUE

grid.xaxis

atNULL

atNULL

[viewportgrid.yaxis](#)

grid.xspline

```
grid.xspline(...)
xsplineGrob(x = c(0, 0.5, 1, 0.5), y = c(0.5, 1, 0.5, 0),
            id = NULL, id.lengths = NULL,
            default.units = "npc",
            shape = 0, open = TRUE, arrow = NULL, repEnds = TRUE,
            name = NULL, gp = gpar(), vp = NULL)
```

```

x
y
id          xyid
id.lengths  xy
default.units xy
shape
open
arrow        arrow
repEnds
name
gp           "gpar" gpar
vp
...          xsplineGrob

```

```
grid.xspline
```

```

repEnds
repEnds
xy

```

<https://dept-info.labri.fr/~schlick/DOC/sig1.html>

```

viewportarrow
xspline

```

```

x <- c(0.25, 0.25, 0.75, 0.75)
y <- c(0.25, 0.75, 0.75, 0.25)

```

```

xsplineTest <- function(s, i, j, open) {
  pushViewport(viewport(layout.pos.col=j, layout.pos.row=i))
  grid.points(x, y, default.units="npc", pch=16, size=unit(2, "mm"))
  grid.xspline(x, y, shape=s, open=open, gp=gpar(fill="grey"))
  grid.text(s, gp=gpar(col="grey"),
    x=unit(x, "npc") + unit(c(-1, -1, 1, 1), "mm"),

```



```

        y=unit(y, "npc") + unit(c(-1, 1, 1, -1), "mm"),
        hjust=c(1, 1, 0, 0),
        vjust=c(1, 0, 0, 1))
    popViewport()
}

pushViewport(viewport(width=.5, x=0, just="left",
    layout=grid.layout(3, 3, respect=TRUE)))
pushViewport(viewport(layout.pos.row=1))
grid.text("Open Splines", y=1, just="bottom")
popViewport()
xsplineTest(c(0, -1, -1, 0), 1, 1, TRUE)
xsplineTest(c(0, -1, 0, 0), 1, 2, TRUE)
xsplineTest(c(0, -1, 1, 0), 1, 3, TRUE)
xsplineTest(c(0, 0, -1, 0), 2, 1, TRUE)
xsplineTest(c(0, 0, 0, 0), 2, 2, TRUE)
xsplineTest(c(0, 0, 1, 0), 2, 3, TRUE)
xsplineTest(c(0, 1, -1, 0), 3, 1, TRUE)
xsplineTest(c(0, 1, 0, 0), 3, 2, TRUE)
xsplineTest(c(0, 1, 1, 0), 3, 3, TRUE)
popViewport()
pushViewport(viewport(width=.5, x=1, just="right",
    layout=grid.layout(3, 3, respect=TRUE)))
pushViewport(viewport(layout.pos.row=1))
grid.text("Closed Splines", y=1, just="bottom")
popViewport()
xsplineTest(c(-1, -1, -1, -1), 1, 1, FALSE)
xsplineTest(c(-1, -1, 0, -1), 1, 2, FALSE)
xsplineTest(c(-1, -1, 1, -1), 1, 3, FALSE)
xsplineTest(c(0, 0, -1, 0), 2, 1, FALSE)
xsplineTest(c(0, 0, 0, 0), 2, 2, FALSE)
xsplineTest(c(0, 0, 1, 0), 2, 3, FALSE)
xsplineTest(c(1, 1, -1, 1), 3, 1, FALSE)
xsplineTest(c(1, 1, 0, 1), 3, 2, FALSE)
xsplineTest(c(1, 1, 1, 1), 3, 3, FALSE)
popViewport()

```

grid.yaxis

```

grid.yaxis(at = NULL, label = TRUE, main = TRUE,
    edits = NULL, name = NULL,
    gp = gpar(), draw = TRUE, vp = NULL)

yaxisGrob(at = NULL, label = TRUE, main = TRUE,
    edits = NULL, name = NULL,
    gp = gpar(), vp = NULL)

```

at

label	at
main	TRUEFALSE
edits	atNULL
name	
gp	"gpar" gpar
draw	
vp	NULL

grid.yaxisdrawTRUE

grid.yaxis

atNULL

atNULL

[viewportgrid.xaxis](#)

gridCoords

[grobPoints](#)

gridCoords(x, y)
gridGrobCoords(x, name, rule = NULL)
gridGTreeCoords(x, name)
emptyCoords
emptyGrobCoords(name)
emptyGTreeCoords(name)
isEmptyCoords(coords)

x	gridCoordsgridGrobCoords"GridCoords"gridGTreeCoords "GridGrobCoords""GridGTreeCoords"
y	
name	
rule	"winding""evenodd"NULL
coords	grobCoords

grobPoints
emptyCoordsclosedisEmptyCoords

gridCoords"GridCoords"gridGrobCoords"GridGrobCoords"gridGTreeCoords
"GridGTreeCoords"

grobCoords

grobCoords(x, closed, ...)
grobPoints(x, closed, ...)
isClosed(x, ...)

x
closed
TRUE
...

grobCoordsgrobPointsgrobCoordsgrobPointsvpgpgrobCoordsgrobPoints
grobPointsgridCoords
isClosedTRUEFALSETRUE

"GridGrobCoords"xy"GridGTreeCoords""GridGrobCoords""GridGTreeCoords"

`grobName`

`grobName(grob = NULL, prefix = "GRID")`

`grob` `NULL`
`prefix`

`prefix.class(grob).index`

`grobWidth`

`grobWidth(x)`
`grobHeight(x)`
`grobAscent(x)`
`grobDescent(x)`

`x`

`unitstringWidth`

`grobX`

```
grobX(x, theta)
grobY(x, theta)
```

```
x
theta      "east""north""west""south"
```

`unitgrobWidth`

`legendGrob`

```
legendGrob(labels, nrow, ncol, byrow = FALSE,
            do.lines = has.lty || has.lwd, lines.first = TRUE,
            hgap = unit(1, "lines"), vgap = unit(1, "lines"),
            default.units = "lines", pch, gp = gpar(), vp = NULL)

grid.legend(..., draw=TRUE)
```

```

labels
nrowncol      nrowncol
byrow
do.lines
lines.first
hgap
vgap
default.units  unit
pch            pointsGrob()points
gp            "gpar"gpar
vp            viewportNULL
...           grid.legend()legendGrob()
draw

```

```
grobgrid.legenddrawTRUE
```

```
viewportpointsGroblinesGrob
```

```
grid.plot.and.legend
```

```

## Data:
n <- 10
x <- stats::runif(n) ; y1 <- stats::runif(n) ; y2 <- stats::runif(n)
## Construct the grobs :
plot <- gTree(children=gList(rectGrob(),
                             pointsGrob(x, y1, pch=21, gp=gpar(col=2, fill="gray")),
                             pointsGrob(x, y2, pch=22, gp=gpar(col=3, fill="gray")),
                             xaxisGrob(),
                             yaxisGrob()))
legd <- legendGrob(c("Girls", "Boys", "Other"), pch=21:23,
                  gp=gpar(col = 2:4, fill = "gray"))
gg <- packGrob(packGrob(frameGrob(), plot),
              legd, height=unit(1,"null"), side="right")

## Now draw it on a new device page:
grid.newpage()
pushViewport(viewport(width=0.8, height=0.8))
grid.draw(gg)

```

makeContent

makeContext(x)
makeContent(x)

x

grid.draw
makeContextvpchildrenvpxxvpchildrenvp
makeContentgrid::makeContent.xaxischildrenxchildrendrawDetails()
xmakeContent()makeContext()
makeContext"groewidth""grobheight"

x

grid.draw

patterns

```

linearGradient(colours = c("black", "white"),
               stops = seq(0, 1, length.out = length(colours)),
               x1 = unit(0, "npc"), y1 = unit(0, "npc"),
               x2 = unit(1, "npc"), y2 = unit(1, "npc"),
               default.units = "npc",
               extend = c("pad", "repeat", "reflect", "none"),
               group = TRUE)
radialGradient(colours = c("black", "white"),
               stops = seq(0, 1, length.out = length(colours)),
               cx1 = unit(.5, "npc"), cy1 = unit(.5, "npc"),
               r1 = unit(0, "npc"),
               cx2 = unit(.5, "npc"), cy2 = unit(.5, "npc"),
               r2 = unit(.5, "npc"),
               default.units = "npc",
               extend = c("pad", "repeat", "reflect", "none"),
               group = TRUE)
pattern(grob,
        x = 0.5, y = 0.5, width = 1, height = 1,
        default.units = "npc",
        just="centre", hjust=NULL, vjust=NULL,
        extend = c("pad", "repeat", "reflect", "none"),
        gp = gpar(fill="transparent"),
        group = TRUE)

```

```

colours
stops
x1y1x2y2
default.units
extend
cx1cy1r1cx2cy2r2

```

```

grob
xywidthheight
justhjustvjust
gp
group

```

```

fillgpar()

```

```

pad
none
repeat
reflect

```

```

extend="repeat"

```

```

group = FALSEgroup = FALSEgroup = FALSE

```


`extend`

`gpar`

`plotViewport`

`plotViewport(margins=c(5.1, 4.1, 4.1, 2.1), ...)`

<code>margins</code>	<code>par(mar)</code>
<code>...</code>	<code>viewport()</code>

`viewportdataViewport`

Querying the Viewport Tree

`current.viewport()`

`current.parent`

`current.vpTree`

`current.vpPath`

`current.transform`

`current.rotation`

`current.viewport()`

`current.parent(n=1)`

`current.vpTree(all=TRUE)`

`current.vpPath()`

`current.transform()`

`current.rotation()`

`n`

`all`

`ncurrent.parent()`

`NULL`

`allFALSEcurrent.vpTree`

`current.viewportcurrent.vpTree`

`current.transform`

`current.vpPathNULLROOT`

[viewport](#)

```
grid.newpage()
pushViewport(viewport(width=0.8, height=0.8, name="A"))
pushViewport(viewport(x=0.1, width=0.3, height=0.6,
  just="left", name="B"))
upViewport(1)
pushViewport(viewport(x=0.5, width=0.4, height=0.8,
  just="left", name="C"))
pushViewport(viewport(width=0.8, height=0.8, name="D"))
current.vpPath()
upViewport(1)
current.vpPath()
current.vpTree()
current.viewport()
current.vpTree(all=FALSE)
popViewport(0)
```

resolveRasterSize

resolveRasterSize(x)

x

NULL

[grid.raster](#)

```
# Square raster
rg <- rasterGrob(matrix(0))
# Fill the complete page (if page is square)
grid.newpage()
resolveRasterSize(rg)$height
grid.draw(rg)
# Forced to fit tall thin region
grid.newpage()
pushViewport(viewport(width=.1))
resolveRasterSize(rg)$height
grid.draw(rg)
```

roundrect

```
roundrectGrob(x=0.5, y=0.5, width=1, height=1,
              default.units="npc",
              r=unit(0.1, "snpc"),
              just="centre",
              name=NULL, gp=NULL, vp=NULL)
grid.roundrect(...)
```

```
xywidthheight
default.units  xywidthheight
r
just
name
gp
vp             NULL
...           roundrectGrob()
```

```
grid.roundrect(width=.5, height=.5, name="rr")
theta <- seq(0, 360, length.out=50)
for (i in 1:50)
  grid.circle(x=grobX("rr", theta[i]),
              y=grobY("rr", theta[i]),
              r=unit(1, "mm"),
              gp=gpar(fill="black"))
```

showGrob

```
showGrob(x = NULL,
         gPath = NULL, strict = FALSE, grep = FALSE,
         recurse = TRUE, depth = NULL,
         labelfun = grobLabel, ...)
```

```

x                NULL
gPath
strict
grep
recurse
depth
labelfun
...             labelfun

```

```
grid.refresh
```

[grobTree](#)

```

grid.newpage()
gt <- gTree(childrenvp=vpStack(
  viewport(x=0, width=.5, just="left", name="vp"),
  viewport(y=.5, height=.5, just="bottom", name="vp2")),
  children=gList(rectGrob(vp="vp:vp2", name="child"),
    name="parent")
grid.draw(gt)
showGrob()
showGrob(gPath="child")
showGrob(recurse=FALSE)
showGrob(depth=1)
showGrob(depth=2)
showGrob(depth=1:2)
showGrob(gt)
showGrob(gt, gPath="child")
showGrob(just="left", gp=gpar(col="red", cex=.5), rot=45)
showGrob(labelfun=function(grob, ...) {
  x <- grobX(grob, "west")
  y <- grobY(grob, "north")
  gTree(children=gList(rectGrob(x=x, y=y,
    width=stringWidth(grob$name) + unit(2, "mm"),
    height=stringHeight(grob$name) + unit(2, "mm"),
    gp=gpar(col=NA, fill=rgb(1, 0, 0, .5)),
    just=c("left", "top")),
    textGrob(grob$name,
      x=x + unit(1, "mm"), y=y - unit(1, "mm"),
      just=c("left", "top"))))
})

## Not run:
# Examples from higher-level packages

library(lattice)
# Ctrl-c after first example
example(histogram)

```

```

showGrob()
showGrob(gPath="plot_01.ylab")

library(ggplot2)
# Ctrl-c after first example
example(qplot)
showGrob()
showGrob(recurse=FALSE)
showGrob(gPath="panel-3-3")
showGrob(gPath="axis.title", grep=TRUE)
showGrob(depth=2)

## End(Not run)

```

showViewport

```

showViewport(vp = NULL, recurse = TRUE, depth = NULL,
             newpage = FALSE, leaves = FALSE,
             col = rgb(0, 0, 1, 0.2), fill = rgb(0, 0, 1, 0.1),
             label = TRUE, nrow = 3, ncol = nrow)

```

vp	NULL
recurse	
depth	
newpage	
leaves	
col	
fill	col
label	
nrowncol	leavesTRUE

[viewportgrid.show.viewport](#)

```

showViewport(viewport(width=.5, height=.5, name="vp"))

grid.newpage()
pushViewport(viewport(width=.5, height=.5, name="vp"))
upViewport()
showViewport(vpPath("vp"))

showViewport(vpStack(viewport(width=.5, height=.5, name="vp1"),
                     viewport(width=.5, height=.5, name="vp2")),

```

```
newpage=TRUE)

showViewport(vpStack(viewport(width=.5, height=.5, name="vp1"),
                      viewport(width=.5, height=.5, name="vp2")),
            fill=rgb(1:0, 0:1, 0, .1),
            newpage=TRUE)
```

stringWidth

```
stringWidth(string)
stringHeight(string)
stringAscent(string)
stringDescent(string)
```

string

unit

unitgrobWidth
strwidth

unit

```
unit(x, units, data=NULL)
is.unit(x)
```

x

is.unit

units

data

unit

```

units
units

"npc"
"cm"
"inches"
"mm"
"points"
"picas"
"bigpts"
"dida"
"cicero"
"scaledpts"
"lines" fontsizelineheight
"char" fontsize
"native" xscaleyscale
"snp"
"strwidth" data
"strheight" data
"grobwidth" data
"grobheight" data

"in" "inch" "inches" "centimetre" "centimeter" "cm"
units"null" grid.layout
dataunit.length()

  unit(rep(1, 3), c("npc", "strwidth", "inches"),
  data = list(NULL, "my string", NULL))

unit.c
unit(1, "npc") - unit(1, "inches")min(unit(0.5, "npc"), unit(1, "inches"))
formatformatdigits
is.unit()x"unit"

"unit"

unit.c
c
"mylines" "mychar" "mystrwidth" "mystrheight" "lines" "char" "strwidth" "strheight"

```


unit.c

```
unit(1, "npc")
unit(1:3/4, "npc")
unit(1:3/4, "npc") + unit(1, "inches")
min(unit(0.5, "npc"), unit(1, "inches"))
unit.c(unit(0.5, "npc"), unit(2, "inches") + unit(1:3/4, "npc"),
        unit(1, "strwidth", "hi there"))
x <- unit(1:5, "npc")
x[2:4]
x[2:4] <- unit(1, "mm")
x
```

unit.c

```
unit.c(..., check = TRUE)
```

```
...
```

```
check          FALSE
```

```
unit
```

unit

`unit.length`

`length`

`unit.length(unit)`

`unit`

`unit`

```
length(unit(1:3, "npc"))
length(unit(1:3, "npc") + unit(1, "inches"))
length(max(unit(1:3, "npc") + unit(1, "inches")))
length(max(unit(1:3, "npc") + unit(1, "strwidth", "a"))*4)
length(unit(1:3, "npc") + unit(1, "strwidth", "a")*4)
```

`unit.pmin`

```
unit.pmin(...)
unit.pmax(...)
unit.psum(...)
```

`...`

```
max(unit(1:3, "cm"), unit(0.5, "npc"))
unit.pmax(unit(1:3, "cm"), unit(0.5, "npc"))
```

unit.rep

timeslength.out

rep

unit.rep(x, ...)

x "unit"

... rep^{timeslength.out}

"unit"

rep

```
rep(unit(1:3, "npc"), 3)
rep(unit(1:3, "npc"), 1:3)
rep(unit(1:3, "npc") + unit(1, "inches"), 3)
rep(max(unit(1:3, "npc") + unit(1, "inches")), 3)
rep(max(unit(1:3, "npc") + unit(1, "strwidth", "a"))*4, 3)
rep(unit(1:3, "npc") + unit(1, "strwidth", "a")*4, 3)
```

unitType

unitType(x, recurse = FALSE)

x

recurse

"inches""npc"

"sum""min""max"

recurse = TRUE

unit

```
u <- unit(1:5, c("cm", "mm", "in", "pt", "null"))

unitType(u)
unitType(unit(1, "npc"))
unitType(unit(1:3/4, "npc"))
unitType(unit(1:3/4, "npc") + unit(1, "inches"))
unitType(min(unit(0.5, "npc"), unit(1, "inches")))
unitType(unit.c(unit(0.5, "npc"), unit(2, "inches") + unit(1:3/4, "npc"),
  unit(1, "strwidth", "hi there")))
unitType(min(unit(1, "in"), unit(1, "npc") + unit(1, "mm"))))

unitType(u, recurse=TRUE)
unitType(unit(1, "npc"), recurse=TRUE)
unitType(unit(1:3/4, "npc"), recurse=TRUE)
unitType(unit(1:3/4, "npc") + unit(1, "inches"), recurse=TRUE)
unitType(min(unit(0.5, "npc"), unit(1, "inches")), recurse=TRUE)
unitType(unit.c(unit(0.5, "npc"), unit(2, "inches") + unit(1:3/4, "npc"),
  unit(1, "strwidth", "hi there")), recurse=TRUE)
unitType(min(unit(1, "in"), unit(1, "npc") + unit(1, "mm")), recurse=TRUE)
unlist(unitType(min(unit(1, "in"), unit(1, "npc") + unit(1, "mm")),
  recurse=TRUE))
```

valid.just

```
valid.just(just)
resolveHJust(just, hjust)
resolveVJust(just, vjust)
```

```
just          "left"
hjust
vjust
```

```
validDetails
```

```
"left""right"
```

validDetails

grobgrid.editeditGrob

validDetails(x)

x

grobgrid.editeditGrobgrid::validDetails.axis
vpgpchildrenchildrenvp

[grid.edit](#)

viewportTransform

grid.define()grid.use()

viewportTransform(group, shear=groupShear(), flip=groupFlip(), device=TRUE)
viewportTranslate(group, device=TRUE)
viewportScale(group, device=TRUE)
viewportRotate(group, device=TRUE)
defnTranslate(group, inverse=FALSE, device=TRUE)
defnScale(group, inverse=FALSE)
defnRotate(group, inverse=FALSE, device=TRUE)
useTranslate(inverse=FALSE, device=TRUE)
useScale(inverse=FALSE)
useRotate(inverse=FALSE, device=TRUE)
groupTranslate(dx=0, dy=0)
groupRotate(r=0, device=TRUE)
groupScale(sx=1, sy=1)
groupShear(sx=0, sy=0)
groupFlip(flipX=FALSE, flipY=FALSE)

```
group
inverse
shear
flip
dxdy
r
sxsy
flipXflipY
device
```

```
viewport*()transformgrid.use
defn*()use*()viewportTransform
group*()groupShear()viewportTransform()
transformgrid.usegroupdevice
```

```
## NOTE: on devices without support for groups nothing will be drawn
grid.newpage()
## Define and use group in same viewport
pushViewport(viewport(width=.2, height=.2))
grid.define(circleGrob(gp=gpar(lwd=5)), name="circle")
grid.use("circle")
popViewport()
## Use group in viewport that is translated and scaled
pushViewport(viewport(x=.2, y=.2, width=.1, height=.1))
grid.use("circle")
popViewport()
## Use group in viewport that is translated and scaled
## BUT only make use of the translation
pushViewport(viewport(x=.2, y=.8, width=.1, height=.1))
grid.use("circle", transform=viewportTranslate)
popViewport()
## Use group in viewport that is translated and scaled
## unevenly (distorted)
pushViewport(viewport(x=.8, y=.7, width=.2, height=.4))
grid.use("circle")
popViewport()
```

vpPath

downViewportseekViewport

vpPath(...)

...

vpPath

[viewportpushViewportpopViewportdownViewportseekViewportupViewport](#)

vpPath("vp1", "vp2")

widthDetails

widthDetails(x)
heightDetails(x)
ascentDetails(x)
descentDetails(x)

x

"grobwidth""grobheight"grid::widthDetails.frame

[absolute.size](#)

Working with Viewports

```
pushViewport(..., recording=TRUE)
popViewport(n = 1, recording=TRUE)
downViewport(name, strict=FALSE, recording=TRUE)
seekViewport(name, recording=TRUE)
upViewport(n = 1, recording=TRUE)
```

```
...          "viewport"
n            0
name
strict
recording
```

```
viewport()
pushViewport()popViewport()
upViewportdownViewportseekViewport
name
```

```
downViewport
depth <- downViewport()upViewport(depth)
```

viewportvpPath

```
# push the same viewport several times
grid.newpage()
vp <- viewport(width=0.5, height=0.5)
pushViewport(vp)
grid.rect(gp=gpar(col="blue"))
grid.text("Quarter of the device",
  y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="blue"))
pushViewport(vp)
grid.rect(gp=gpar(col="red"))
```



```

grid.text("Quarter of the parent viewport",
    y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="red"))
popViewport(2)
# push several viewports then navigate amongst them
grid.newpage()
grid.rect(gp=gpar(col="grey"))
grid.text("Top-level viewport",
    y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="grey"))
if (interactive()) Sys.sleep(1.0)
pushViewport(viewport(width=0.8, height=0.7, name="A"))
grid.rect(gp=gpar(col="blue"))
grid.text("1. Push Viewport A",
    y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="blue"))
if (interactive()) Sys.sleep(1.0)
pushViewport(viewport(x=0.1, width=0.3, height=0.6,
    just="left", name="B"))
grid.rect(gp=gpar(col="red"))
grid.text("2. Push Viewport B (in A)",
    y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="red"))
if (interactive()) Sys.sleep(1.0)
upViewport(1)
grid.text("3. Up from B to A",
    y=unit(1, "npc") - unit(2, "lines"), gp=gpar(col="blue"))
if (interactive()) Sys.sleep(1.0)
pushViewport(viewport(x=0.5, width=0.4, height=0.8,
    just="left", name="C"))
grid.rect(gp=gpar(col="green"))
grid.text("4. Push Viewport C (in A)",
    y=unit(1, "npc") - unit(1, "lines"), gp=gpar(col="green"))
if (interactive()) Sys.sleep(1.0)
pushViewport(viewport(width=0.8, height=0.6, name="D"))
grid.rect()
grid.text("5. Push Viewport D (in C)",
    y=unit(1, "npc") - unit(1, "lines"))
if (interactive()) Sys.sleep(1.0)
upViewport(0)
grid.text("6. Up from D to top-level",
    y=unit(1, "npc") - unit(2, "lines"), gp=gpar(col="grey"))
if (interactive()) Sys.sleep(1.0)
downViewport("D")
grid.text("7. Down from top-level to D",
    y=unit(1, "npc") - unit(2, "lines"))
if (interactive()) Sys.sleep(1.0)
seekViewport("B")
grid.text("8. Seek from D to B",
    y=unit(1, "npc") - unit(2, "lines"), gp=gpar(col="red"))
pushViewport(viewport(width=0.9, height=0.5, name="A"))
grid.rect()
grid.text("9. Push Viewport A (in B)",
    y=unit(1, "npc") - unit(1, "lines"))
if (interactive()) Sys.sleep(1.0)
seekViewport("A")
grid.text("10. Seek from B to A (in ROOT)",
    y=unit(1, "npc") - unit(3, "lines"), gp=gpar(col="blue"))
if (interactive()) Sys.sleep(1.0)
seekViewport(vpPath("B", "A"))
grid.text("11. Seek from nA (in ROOT) into A (in B)")

```

popViewport(0)

xDetails

xDetails(x, theta)
yDetails(x, theta)

x
theta "north""east""west""south"

theta
"grobx""groby"grobxgroby

grobXgrobY

xsplinePoints

xsplinePoints(x)
bezierPoints(x)

x xsplineGrob()bezierGrob()

xsplineGrobbezierGrob

```
grid.newpage()
xsg <- xsplineGrob(c(.1, .1, .9, .9), c(.1, .9, .9, .1), shape=1)
grid.draw(xsg)
trace <- xsplinePoints(xsg)
grid.circle(trace$x, trace$y, default.units="inches", r=unit(.5, "mm"))

grid.newpage()
vp <- viewport(width=.5)
xg <- xsplineGrob(x=c(0, .2, .4, .2, .5, .7, .9, .7),
                  y=c(.5, 1, .5, 0, .5, 1, .5, 0),
                  id=rep(1:2, each=4),
                  shape=1,
                  vp=vp)
grid.draw(xg)
trace <- xsplinePoints(xg)
pushViewport(vp)
invisible(lapply(trace, function(t) grid.lines(t$x, t$y, gp=gpar(col="red"))))
popViewport()

grid.newpage()
bg <- bezierGrob(c(.2, .2, .8, .8), c(.2, .8, .8, .2))
grid.draw(bg)
trace <- bezierPoints(bg)
grid.circle(trace$x, trace$y, default.units="inches", r=unit(.5, "mm"))
```

methods

methods-package

[setClasssetMethod](#)

[setClassgetClassDefisas](#)

[setGeneric](#)

[setOldClass](#)

[setClassUnion](#)

`library(help="methods")`

<R-core@r-project.org>

`.BasicFunsList`

`.Primitive(<name>)`

`for.Primitive`
`setMethod.BasicFunsList(x, ...)`

`as`

`as(object, Class, strict=TRUE, ext)`

`as(object, Class) <- value`

<code>object</code>	
<code>Class</code>	<code>object</code>
<code>strict</code>	<code>TRUE</code>
	<code>strict = FALSE</code>
<code>value</code>	<code>objectClass</code>
<code>ext</code>	<code>ClasspossibleExtends</code>

`as(object)ClassClassvalue`
`as()`

`setAsshowMethods(coerce)`

`as(x, "numeric")as.numeric`

`try(as(x, cl))canCoerce(x, cl)`

`## Show all the existing methods for as()`
`showMethods("coerce")`

BasicClasses

```
### The following are all basic vector classes.
### They can appear as class names in method signatures,
### in calls to as(), is(), and new().
"character"
"complex"
"double"
"expression"
"integer"
"list"
"logical"
"numeric"
"single"
"raw"

### the class
"vector"
### is a virtual class, extended by all the above

### the class
"S4"
### is an object type for S4 objects that do not extend
### any of the basic vector classes. It is a virtual class.

### The following are additional basic classes
"NULL"      # NULL objects
"function"  # function objects, including primitives
"externalptr" # raw external pointers for use in C code

"ANY" # virtual classes used by the methods package itself
"VIRTUAL"
"missing"

"namedList" # the alternative to "list" that preserves
             # the names attribute

Classes_Detailsnew(Class, ...)Class
"expression"quote()
"list"NULL"namedList""list""names""character""namedList"NULL

isS4FALSEtypeof"symbol""name""language""language"
```

"vector"

as(x, "numeric")as.numeric(x)

callGeneric

callGenericcallGeneric

callGeneric(...)

...
callGeneric

callGeneric"missing"xx = x
callGeneric

GroupGenericFunctions

```
## the method for group generic function Ops
## for signature( e1="structure", e2="vector")
function (e1, e2)
{
  value <- callGeneric(e1@.Data, e2)
  if (length(value) == length(e1)) {
    e1@.Data <- value
    e1
  }
  else value
}

## For more examples
## Not run:
showMethods("Ops", includeDefs = TRUE)

## End(Not run)
```

`callNextMethod`

`callNextMethodcallNextMethod``callNextMethod(...)``...``callNextMethod``callNextMethod``definedcallNextMethodselectMethod``callNextMethod()``"missing"xx = x``callNextMethod()``"rnum"callNextMethod``callNextMethod``callGeneric``## callNextMethod() used for the Math, Math2 group generic functions``## A class to automatically round numeric results to "d" digits``rnum <- setClass("rnum", slots = c(d = "integer"), contains = "numeric")``## Math functions operate on the rounded numbers, return a plain
vector. The next method will always be the default, usually a primitive.
setMethod("Math", "rnum",``function(x)``callNextMethod(round(as.numeric(x), x@d)))`


```

setMethod("Math2", "rnum",
  function(x, digits)
    callNextMethod(round(as.numeric(x), x@d), digits))

## Examples of callNextMethod with two arguments in the signature.

## For arithmetic and one rnum with anything, callNextMethod with no arguments
## round the full accuracy result, and return as plain vector
setMethod("Arith", c(e1 = "rnum"),
  function(e1, e2)
    as.numeric(round(callNextMethod(), e1@d)))
setMethod("Arith", c(e2 = "rnum"),
  function(e1, e2)
    as.numeric(round(callNextMethod(), e2@d)))

## A method for BOTH arguments from "rnum" would be ambiguous
## for callNextMethod(): the two methods above are equally valid.
## The method chooses the smaller number of digits,
## and then calls the generic function, postponing the method selection
## until it's not ambiguous.
setMethod("Arith", c(e1 = "rnum", e2 = "rnum"),
  function(e1, e2) {
    if(e1@d <= e2@d)
      callGeneric(e1, as.numeric(e2))
    else
      callGeneric(as.numeric(e1), e2)
  })

## For comparisons, callNextMethod with the rounded arguments
setMethod("Compare", c(e1 = "rnum"),
  function(e1, e2)
    callNextMethod(round(e1, e1@d), round(e2, e1@d)))
setMethod("Compare", c(e2 = "rnum"),
  function(e1, e2)
    callNextMethod(round(e1, e2@d), round(e2, e2@d)))

## similarly to the Arith case, the method for two "rnum" objects
## can not unambiguously use callNextMethod(). Instead, we rely on
## The rnum() method inherited from Math2 to return plain vectors.
setMethod("Compare", c(e1 = "rnum", e2 = "rnum"),
  function(e1, e2) {
    d <- min(e1@d, e2@d)
    callGeneric(round(e1, d), round(e2, d))
  })

set.seed(867)

x1 <- rnum(10*runif(5), d=1L)
x2 <- rnum(10*runif(5), d=2L)

x1+1
x2*2
x1-x2

```

```

## Simple examples to illustrate callNextMethod with and without arguments
B0 <- setClass("B0", slots = c(s0 = "numeric"))

## and a function to illustrate callNextMethod

f <- function(x, text = "default") {
  str(x) # print a summary
  paste(text, ":", class(x))
}

setGeneric("f")
setMethod("f", "B0", function(x, text = "B0") {
  cat("B0 method called with s0 =", x@s0, "\n")
  callNextMethod()
})

b0 <- B0(s0 = 1)

## call f() with 2 arguments: callNextMethod passes both to the default method
f(b0, "first test")

## call f() with 1 argument: the default "B0" is not passed by callNextMethod
f(b0)

## Now, a class that extends B0, with no methods for f()
B1 <- setClass("B1", slots = c(s1 = "character"), contains = "B0")
b1 <- B1(s0 = 2, s1 = "Testing B1")

## the two cases work as before, by inheriting the "B0" method

f(b1, b1@s1)

f(b1)

B2 <- setClass("B2", contains = "B1")

## And, a method for "B2" that calls with explicit arguments.
## Note that the method selection in callNextMethod
## uses the class of the *argument* to consistently select the "B0" method

setMethod("f", "B2", function(x, text = "B1 method") {
  y <- B1(s0 = -x@s0, s1 = "Modified x")
  callNextMethod(y, text)
})

b2 <- B2(s1 = "Testing B2", s0 = 10)

f(b2, b2@s1)

f(b2)

## Be careful: the argument passed must be legal for the method selected
## Although the argument here is numeric, it's still the "B0" method that's called
setMethod("f", "B2", function(x, text = "B1 method") {
  callNextMethod(x@s0, text)
})

```

```
## Now the call will cause an error:

tryCatch(f(b2), error = function(e) cat(e$message, "\n"))
```

canCoerce

```
if()as(object, Class)
```

```
canCoerce(object, Class)
```

```
object
```

```
Class          isClass
```

```
TRUEcoercesetAs(from = class(object), to = Class)
```

```
assetAsselectMethodsetClass
```

```
m <- matrix(pi, 2,3)
canCoerce(m, "numeric") # TRUE
canCoerce(m, "array")   # TRUE
```

cbind2

```
cbind2rbind2
```

```
cbind2(x, y, ...)
rbind2(x, y, ...)
```

```
x
```

```
y          x
```

```
...
```

```
cbind2rbind2cbind()rbind()
```

```
cbind2rbind2...cbindrbind  
cbindrbinddata.framecbind2rbind2
```

```
xycolnamesrownamesxycbind(..., deparse.level = d) $d \geq 1$ 
```

```
signature(x = "ANY", y = "ANY")  
signature(x = "ANY", y = "missing")
```

[cbindrbind](#)

```
cbind2(1:3, 4)  
m <- matrix(3:8, 2,3, dimnames=list(c("a","b"), LETTERS[1:3]))  
cbind2(1:2, m) # keeps dimnames from m  
  
## rbind() and cbind() now make use of rbind2()/cbind2() methods  
setClass("Num", contains="numeric")  
setMethod("cbind2", c("Num", "missing"),  
  function(x,y, ...) { cat("Num-miss--meth\n"); as.matrix(x)})  
setMethod("cbind2", c("Num","ANY"), function(x,y, ...) {  
  cat("Num-A.--method\n") ; cbind(getDataPart(x), y, ...) })  
setMethod("cbind2", c("ANY","Num"), function(x,y, ...) {  
  cat("A.-Num--method\n") ; cbind(x, getDataPart(y), ...) })  
  
a <- new("Num", 1:3)  
trace("cbind2")  
cbind(a)  
cbind(a, four=4, 7:9)# calling cbind2() twice  
  
cbind(m,a, ch=c("D","E"), a*3)  
cbind(1,a, m) # ok with a warning  
untrace("cbind2")
```

Classes

[setClass](#)

classesToAM

contains

```
classesToAM(classes, includeSubclasses = FALSE,  
            abbreviate = 2)
```

```
classes  
includeSubclasses  
            TRUEclasses  
abbreviate
```

extends

```
## the super- and subclasses of "standardGeneric"  
## and "derivedDefaultMethod"  
am <- classesToAM(list(class(show), class(getMethod(show))), TRUE)  
am  
  
## Not run:  
## the following function depends on the Bioconductor package Rgraphviz  
plotInheritance <- function(classes, subclasses = FALSE, ...) {  
  if(!require("Rgraphviz", quietly=TRUE))  
    stop("Only implemented if Rgraphviz is available")  
  mm <- classesToAM(classes, subclasses)  
  classes <- rownames(mm); rownames(mm) <- colnames(mm)  
  graph <- new("graphAM", mm, "directed", ...)  
  plot(graph)  
  cat("Key:\n", paste(abbreviate(classes), " = ", classes, ", ", "  
    sep = """, sep = """, fill = TRUE)  
  invisible(graph)  
}  
  
## The plot of the class inheritance of the package "graph"  
require(graph)  
plotInheritance(getClasses("package:graph"))  
  
## End(Not run)
```

Classes_Details

classRepresentation

getClass

```
slot() "@"
is
.Datamatrixarraycontains.xData"environment"
names"names""namedList"names()"names"names()
FancySimpleFancySimpleSimpleFancy
SimpleFancyFancySimple
contains=setClasssetIscontains=

contains=setClass
SClassExtensionextendsfullInfo=TRUE
newnewinitialize
NULL

"numeric""logical""integer""complex""character""raw""list""expression"
"function""call""language""environment""externalptr"

contains=setClass

setOldClassgetClass
environment
matrixarraydimdimnames

UseMethodmatrixarray
"myFrame""data.frame"contains=setClassas.matrix"data.frame"UseMethod"matrix"
"array"
asS4

matrixarraysetOldClass
.Data.Datatypeof(x).DatagetDataPartsetDataPart
.Data"NULL".xData"simple" is
```

Methods_DetailssetClassisasnewslot

className

```
className() "className" setMethod  
"multipleClasses"
```

```
className(class, package)  
multipleClasses(details = FALSE)
```

```
classpackage    packageclass "className"
```

```
details          FALSEmultipleClasses()  
                  TRUE
```

```
setOldClass
```

```
className() "className"  
multipleClasses() 0
```

```
"className""character""package""character"
```

```
## Not run:  
className("vector") # will be found, from package "methods"  
className("vector", "magic") # OK, even though the class doesn't exist
```

```
className("An unknown class") # Will cause an error
```

```
## End(Not run)
```

classRepresentation-class

[setClass](#)

[setIs](#)

"classRepresentation"[getClass](#)[getClassDef](#)
containssubclasses

slots

contains [SClassExtension](#)

virtual TRUE

prototype [new](#)

validity [validObject](#)

access

className

package

subclasses [SClassExtension](#)

versionKey "externalptr"

sealed "logical"

[setClass](#)

Documentation

[?help](#)

class?

class ? [genericFunction](#)

"genericFunction"

?methods

methods ? [initialize](#)

[initialize](#)


```
"?"  
myFun(x, sqrt(wt))?"  
?myFun(x, sqrt(wt))  
myFun  
"maybeNumber""logical""?"method  
method ? myFun("maybeNumber", "logical")
```

```
"maybeNumber"setClassUnion  
selectMethod
```

```
promptMethods("myFun")  
myFun-methods.RdmyFunpromptMethodsman  
methods ? myFun  
promptMethods  
  \aliasMethodspromptMethods  
  \alias\itemMethods\alias\alias{myfun-methods}
```

dotsMethods	...
-------------	-----

```
setMethod
```

```
setClassUnion
```

```
setMethod
```

```
contains
```

standardGeneric

[showMethods](#)

```
cc <- function(...)c(...)

setGeneric("cc")

setMethod("cc", "character", function(...)paste(...))

setClassUnion("Number", c("numeric", "complex"))

setMethod("cc", "Number", function(...) sum(...))

setClass("cdate", contains = "character", slots = c(date = "Date"))

setClass("vdate", contains = "vector", slots = c(date = "Date"))

cd1 <- new("cdate", "abcdef", date = Sys.Date())

cd2 <- new("vdate", "abcdef", date = Sys.Date())

stopifnot(identical(cc(letters, character(), cd1),
  paste(letters, character(), cd1))) # the "character" method

stopifnot(identical(cc(letters, character(), cd2),
  c(letters, character(), cd2)))
# the default, because "vdate" doesn't extend "character"

stopifnot(identical(cc(1:10, 1+1i), sum(1:10, 1+1i))) # the "Number" method

stopifnot(identical(cc(1:10, 1+1i, TRUE), c(1:10, 1+1i, TRUE))) # the default

stopifnot(identical(cc(), c())) # no arguments implies the default method

setGeneric("numMax", function(...)standardGeneric("numMax"))

setMethod("numMax", "numeric", function(...)max(...))
# won't work for complex data
setMethod("numMax", "Number", function(...) paste(...))
# should not be selected w/o complex args

stopifnot(identical(numMax(1:10, pi, 1+1i), paste(1:10, pi, 1+1i)))
stopifnot(identical(numMax(1:10, pi, 1), max(1:10, pi, 1)))
```

```

try(numMax(1:10, pi, TRUE)) # should be an error: no default method

## A generic version of paste(), dispatching on the "..." argument:
setGeneric("paste", signature = "...")

setMethod("paste", "Number", function(..., sep, collapse) c(...))

stopifnot(identical(paste(1:10, pi, 1), c(1:10, pi, 1)))

```

environment-class	"environment"
-------------------	---------------

```

new("environment", ...)

signature(from = "ANY", to = "environment")as.environment
signature(object = "environment")object

new.env

```

envRefClass-class	"envRefClass"
-------------------	---------------

```

setRefClasssetRefClass$

setRefClass
$$<-initialize
envRefClass
$$
"fieldAccessorGenerator"setRefClassenvRefClass

is(x, "refClass")xis(x, "refObject")"environment"
"classMethodDefinition"

```

evalSource

[trace](#)

```
evalSource(source, package = "", lock = TRUE, cache = FALSE)
```

```
insertSource(source, package = "", functions = , methods = ,  
             force = )
```

```
source      evalSource  
            insertSource"sourceEnvironment"evalSourceinsertSourceevalSource  
package  
functionsmethods  
            functionsmethodssetMethodinsertSource  
            what  
lockcache   evalSourcelockTRUEcacheFALSEsource  
            lockinsertSource  
force       FALSEtraceTRUETRUEfunctionsmethods
```

```
source
```

[untrace](#)[trace](#)

```
functionsmethodsinsertSourcesource
```

```
source
```

```
"sourceEnvironment""environment"  
sourceinsertSourcefunctionsmethods
```

```
insertSourcedebugtrace  
traceedit = envenvevalSource  
trace(x,edit = env)untrace(x)
```

```
"sourceEnvironment"
```

```
evalSource  
packageName  
dateCreated Sys.time  
sourceFile  
            dateCreated
```

`traceedit=debugsetBreakpoint`

```
## Not run:
## Suppose package P0 has a source file "all.R"
## First, evaluate the source, and from it
## insert the revised version of methods for summary()
env <- insertSource("./P0/R/all.R", package = "P0",
  methods = "summary")
## now test one of the methods, tracing the version from the source
trace("summary", signature = "myMat", browser, edit = env)
## After testing, remove the browser() call but keep the source
trace("summary", signature = "myMat", edit = env)
## Now insert all the (other) revised functions and methods
## without re-evaluating the source file.
## The package name is included in the object env.
insertSource(env)

## End(Not run)
```

findClass

`isClass``findClass``getClasses``getClass`

`isClass(Class, formal=TRUE, where)`

`getClasses(where, inherits = missing(where))`

`findClass(Class, where, unique = "")`

The remaining functions are retained for compatibility
but not generally recommended

`removeClass(Class, where)`

`resetClass(Class, classDef, where)`

`sealClass(Class, where)`

<code>Class</code>	<code>packageSlot</code>
<code>where</code>	<code>environment</code>
	<code>where = asNamespace(pkg)pkgwhere = "package:pkg"</code>
<code>formal</code>	<code>logicalTRUE</code>
<code>unique</code>	<code>findClassunique</code>
<code>inherits</code>	<code>getClasseswhereTRUEwhereFALSE</code>
<code>classDef</code>	<code>resetClass</code>

```

isClass
getClasses wherever
findClass Classwhere
  uniquefindClass

removeClass
resetClass
sealClass

```

[getClassClasses_DetailsMethods_DetailsmakeClassRepresentation](#)

findMethods

```

findMethodslistOfMethods
whereclasses
environmentgetMethodsForDispatchfindMethods
findMethodSignaturesfindMethodsfindMethods
hasMethodsTRUEFALSEfwhere
getMethodstable=FALSEfindMethodsMethodsList

findMethods(f, where, classes = character(), inherited = FALSE,
  package = "")

findMethodSignatures(..., target = TRUE, methods = )

hasMethods(f, where, package)

## Deprecated in 2010 and defunct in 2015 for 'table = FALSE':
getMethod(f, where, table = FALSE) # deprecated -> use getMethodsForDispatch()

f
where
  wherefindMethodshasMethods
table      TRUEgetMethodsmList
classes
inherited  TRUETRUEwhere
...        findMethodSignaturesfindMethods
target     findMethodSignaturesTRUEFALSEinheritedTRUE
methods    findMethodSignaturesfindMethods
package    hasMethods"base""package"

```

```
wheregetMethodstable  
hasMethodspackagef
```

```
"listOfMethods""#"
```

```
.Data "list"  
  findMethodSignatures  
arguments "character"  
signatures "list"names"#"  
  findMethodsfindMethodSignatures  
generic "genericFunction"  
names "character""#"
```

```
"namedList"  
"list""namedList"  
"vector""namedList"
```

```
showMethodsselectMethod
```

```
mm <- findMethods("Ops")  
findMethodSignatures(methods = mm)
```

```
fixPre1.8
```

```
fixPre1.8
```

```
fixPre1.8(names, where)
```

```
names  
where      fixPre1.8
```

fixPre1.8

genericFunction-class

genericFunction

[setGenericsetGroupGenericsetMethod](#)

[setGenericsetGroupGeneric](#)"genericFunction""groupGenericFunction"

.Data "function"[standardGeneric](#)

generic "character"

package "character"setGeneric

group "list"

valueClass "character"

signature "character"

default "optionalMethod""function""NULL"

skeleton "call"

"function"

"optionalMethod""PossibleMethod""OptionalFunction""function"

GenericFunctions

```
isGeneric(f, where, fdef, getName = FALSE)
isGroup(f, where, fdef)
removeGeneric(f, where)

dumpMethod(f, signature, file, where, def)
findFunction(f, generic = TRUE, where = toplevel(parent.frame()))
dumpMethods(f, file, signature, methods, where)
signature(...)

removeMethods(f, where = toplevel(parent.frame()), all = missing(where))

setReplaceMethod(f, ..., where = toplevel(parent.frame()))

getGenerics(where, searchForm = FALSE)
```

```
f
where
signature      setMethod
                signaturedumpMethods

file
def
...
generic        FALSE
fdef
                isGeneric

getName        TRUEisGenericTRUE
methods        where
all            removeMethods
searchForm      getGenericsTRUEpackagesearch\(\)"package:base""base"
```

```
isGeneric fdefNULLfwhere
  fdefNULLff
  getNameTRUEFALSE
  isGenericgetGenericsisGenericgetGeneric
removeGenericremoveMethods removeGenericremoveMethodsremoveMethods
standardGeneric fstandardGeneric
  standardGeneric
```

```

dumpMethod
findFunction name
    findmode="function"findFunctionlibrary
dumpMethods
signature
getGenerics where
    "initialize"package"base""package:base"searchForm

```

```

isGeneric fdeff
removeGeneric where
standardGeneric standardGeneric
    setGenericsetMethodstandardGeneric
dumpMethod
findFunction genericFALSE
dumpMethods signature
signature signaturesignature
removeMethods TRUEfFALSE
    setMethod

```

```

getMethodselectMethodsetGenericsetClassshowMethods

```

```

require(stats) # for lm

## get the function "myFun" -- throw an error if 0 or > 1 versions visible:
findFuncStrict <- function(fName) {
  allF <- findFunction(fName)
  if(length(allF) == 0)
    stop("No versions of ",fName," visible")
  else if(length(allF) > 1)
    stop(fName," is ambiguous: ", length(allF), " versions")
  else
    get(fName, allF[[1]])
}

try(findFuncStrict("myFun"))# Error: no version
lm <- function(x) x+1
try(findFuncStrict("lm"))# Error: 2 versions
findFuncStrict("findFuncStrict")# just 1 version
rm(lm)

```

```

## method dumping -----

```

```

setClass("A", slots = c(a="numeric"))
setMethod("plot", "A", function(x,y,...){ cat("A meth\n") })
dumpMethod("plot","A", file="")
## Not run:
setMethod("plot", "A",
function (x, y, ...)
{
    cat("AAAAA\n")
}
)

## End(Not run)
tmp <- tempfile()
dumpMethod("plot","A", file=tmp)
## now remove, and see if we can parse the dump
stopifnot(removeMethod("plot", "A"))
source(tmp)
stopifnot(is(getMethod("plot", "A"), "MethodDefinition"))

## same with dumpMethods() :
setClass("B", contains="A")
setMethod("plot", "B", function(x,y,...){ cat("B ... \n") })
dumpMethods("plot", file=tmp)
stopifnot(removeMethod("plot", "A"),
          removeMethod("plot", "B"))
source(tmp)
stopifnot(is(getMethod("plot", "A"), "MethodDefinition"),
          is(getMethod("plot", "B"), "MethodDefinition"))

```

getClass

```

getClass (Class, .Force = FALSE, where)
getClassDef(Class, where, package, inherits = TRUE)

```

Class	"package"package
.Force	TRUENULL
where	
package	whereClassClassclass(x)
inherits	FALSEwhere

```

getClassDefinheritsFALSEpackagewhere

```

```
getClassDefNULLgetClassgetClassDef.ForceTRUE
```

```
classRepresentation
```

```
setClasssetClassUnion
```

```
setClassisClass
```

```
getClass("numeric") ## a built in class
```

```
cld <- getClass("thisIsAnUndefinedClass", .Force = TRUE)
```

```
cld ## a NULL prototype
```

```
## If you are really curious:
```

```
utils::str(cld)
```

```
## Whereas these generate errors:
```

```
try(getClass("thisIsAnUndefinedClass"))
```

```
try(getClassDef("thisIsAnUndefinedClass"))
```

```
getMethod
```

```
selectMethod()fsignatureoptional = TRUENULL
```

```
findMethod()findMethod()
```

```
getMethod()selectMethod
```

```
hasMethod()existsMethod()selectMethod()getMethod()
```

```
selectMethod(f, signature, optional = FALSE, useInherited = ,  
             mlist = , fdef = , verbose = , doCache = )
```

```
findMethod(f, signature, where)
```

```
getMethod(f, signature = character(), where, optional = FALSE,  
          mlist, fdef)
```

```
existsMethod(f, signature = character(), where)
```

```
hasMethod(f, signature = character(), where)
```

```
f
```

```
signature      f
```

```
where          findMethod
```

```
optional       selectMethodoptionalTRUENULL
```

```
mlistfdefuseInheritedverbosedoCache
```

```
getMethodselectMethodas()
```

```

signature
signature
match.call
"missing" "ANY" "ANY"
getMethod
selectMethod
getMethod
selectMethod
selectMethod
NULL
optional
selectMethod
getMethod
findMethod
where=findMethod()
existsMethod
hasMethod
TRUE
FALSE
selectMethod
selectMethod

selectMethod
getMethod
function
optional
FALSE
NULL
optional
TRUE
MethodDefinition
is.null()
findMethod

findMethod()
find
findMethod
exportMethods

thisPkg
findMethod(f, signature, where = asNamespace("thisPkg"))
findMethod
where
thisPkg
"ANY"

as()

setAs
coerce(from, to)
useInherited = c(TRUE, FALSE)
selectMethod

```

Methods_DetailsGenericFunctionsMethodDefinition

```

testFun <- function(x)x
setGeneric("testFun")
setMethod("testFun", "numeric", function(x)x+1)

hasMethod("testFun", "numeric") # TRUE

hasMethod("testFun", "integer") #TRUE, inherited

existsMethod("testFun", "integer") #FALSE

hasMethod("testFun") # TRUE, default method

hasMethod("testFun", "ANY")

```

getPackageName

```
getPackageName(where, create = TRUE)  
setPackageName(pkg, env)
```

```
packageSlot(object)  
packageSlot(object) <- value
```

where

object

value

create TRUE

pkgenv pkgenv

[library](#).packageName

getPackageName"package:"

packageSlot

setPackageName[package.skeleton](#)

[search](#)packageName

```
## all the following usually return "base"  
getPackageName(length(search()))  
getPackageName(baseenv())  
getPackageName(asNamespace("base"))  
getPackageName("package:base")
```

hasArg

TRUEname...FALSE

hasArg(name)

name

hasArg(x)!missing(x)hasArgxx...hasArgmissing(x)x

TRUEFALSE

[missing](#)

```
fctest <- function(x1, ...) c(hasArg(x1), hasArg("y2"))

fctest(1) ## c(TRUE, FALSE)
fctest(1, 2) ## c(TRUE, FALSE)
fctest(y2 = 2) ## c(FALSE, TRUE)
fctest(y = 2) ## c(FALSE, FALSE) (no partial matching)
fctest(y2 = 2, x = 1) ## c(TRUE, TRUE) partial match x1
```

implicitGeneric

[setMethodsetGeneric](#)signaturevalueClass

implicitGeneric()setGenericImplicit()prohibitGeneric()
registerImplicitGenerics()

implicitGeneric(name, where, generic)
setGenericImplicit(name, where, restore = TRUE)
prohibitGeneric(name, where)
registerImplicitGenerics(what, where)

name

where

generic

restore

what

```
plot()
setGenericImplicit

with()expr
prohibitGeneric()
implicitGeneric()setGeneric(name)
```

```
implicitGeneric()
```

```
setGeneric()signatureimplicitGeneric()
```

```
setGeneric
```

```
### How we would make the function with() into a generic:

## Since the second argument, 'expr' is used literally, we want
## with() to only have "data" in the signature.

## Not run:
setGeneric("with", signature = "data")
## Now we could predefine methods for "with" if we wanted to.

## When ready, we store the generic as implicit, and restore the
## original

setGenericImplicit("with")

## End(Not run)

implicitGeneric("with")

# (This implicit generic is stored in the 'methods' package.)
```

```
inheritedSlotNames
```

```
getClassclassRepresentation
```

```
inheritedSlotNames(Class, where = topenv(parent.frame()))
```



```
Class      classRepresentationgetClass
where      isClassgetClass
```

```
NULL
```

```
slotNameslotsetClass
```

```
.srch <- search()
library(stats4)
inheritedSlotNames("mle")

if(require("Matrix", quietly = TRUE)) withAutoprint({
  inheritedSlotNames("Matrix")      # NULL
  ## whereas
  inheritedSlotNames("sparseMatrix") # --> Dim & Dimnames
  ## i.e. inherited from "Matrix" class
  cl <- getClass("dgCMatrix")       # six slots, etc
  inheritedSlotNames(cl) # *all* six slots are inherited
})
## Not run:
## detach package we've attached above:
for(n in rev(which(is.na(match(search(), .srch)))))
  try( detach(pos = n) )

## End(Not run)
```

```
initialize-methods
```

```
newinitialize
```

```
signature(.Object = "ANY") initializeas(object, Class, strict = FALSE)
```

```
signature(.Object = "traceable") traceabletrace
  initializedef, tracer, exit, at, printtrace
signature(.Object = "environment")signature(.Object = ".environment")
  initialize"environment".environment"callNextMethod
signature(.Object = "signature") functionDefsignature
```

```

initialize.Objectinitializenew.Object
initializecallNextMethod
.Object"traceable"setMethod

    setMethod("initialize", "traceable",
        function(.Object, def, tracer, exit, at, print) { .... }
    )

xx

function(.Object, x, ...) {
  Object <- callNextMethod(.Object, ...)
  if(!missing(x)) { # do something with x

x

```

Introduction

`setClass()``setMethod`

`setClass()`

```

Pos <- setClass("Pos", slots = c(latitude = "numeric", longitude = "numeric",
altitude = "numeric"))

```

```

Pos(latitude = x, longitude = y, altitude = z)

```

```

@gg@latitude

```

`setClass()`

```

POSexportClasses()export()NAMESPACE

```

```

exportClasses(Pos); export(Pos)

```

"Pos"

```

setMethod("plot", c("Pos", "missing"), function(x, y, ...) { plotPos(x, y) })

```

`plot()`"Pos""ANY"

```

contains =setClass()
"Pos""GPS"
GPS <- setClass("GPS", slots = c(time = "POSIXt"), contains = "Pos")
setOldClass()"POSIXt"
setOldClass("POSIXt")
Currency"unit"
Currency <- setClass("Currency", slots = c(unit = "character"), contains =
"numeric")
"numeric""S4"

```

is

extends

is(object, class2)

extends(class1, class2, maybe = TRUE, fullInfo = FALSE)

object

class1class2 is

fullInfo extendsclass2fullInfoTRUESClassExtension

maybe

[selectSuperClasses](#)(cl)extends(cl)

[sapply](#)

extendsextendsfullInfo = TRUESClassExtension"distance"

extends[sapply](#)TRUE

function(what) is(what, "SClassExtension") && what@distance == 2

"myPkg"

function(what) getClassDef(what)@package == "myPkg"

[sapply](#)extends

extendsclass1TRUE

class1class2

[inheritsissetIs](#)

```
## Not run:
## this example can be run if package XRPYthon from CRAN is installed.
supers <- extends("PythonInterface")
## find all the superclasses from package XR
fromXR <- sapply(supers,
  function(what) getClassDef(what)@package == "XR")
## print them
supers[fromXR]

## find all the superclasses at distance 2
superRelations <- extends("PythonInterface", fullInfo = TRUE)
dist2 <- sapply(superRelations,
  function(what) is(what, "SClassExtension") && what@distance == 2)
## print them
names(superRelations)[dist2]

## End(Not run)
```

[isSealedMethod](#)

```
isSealedMethod(f, signature, fdef, where)
isSealedClass(Class, where)

f
signature      setMethod
fdef           f
Class
where          isSealedMethodisSealedClass
```

[sealedsetClasssetMethod](#)

FALSETRUE

```
## these are both TRUE
isSealedMethod("+", c("numeric", "character"))
isSealedClass("matrix")

setClass("track", slots = c(x="numeric", y="numeric"))
## but this is FALSE
isSealedClass("track")
## and so is this
isSealedClass("A Name for an undefined Class")
## and so are these, because only one of the two arguments is basic
isSealedMethod("+", c("track", "numeric"))
isSealedMethod("+", c("numeric", "track"))
```

language-class

"language"[quote](#)

```
### each of these classes corresponds to an unevaluated object
### in the S language.
### The class name can appear in method signatures,
### and in a few other contexts (such as some calls to as()).
```

```
"("
"<-"
"call"
"for"
"if"
"repeat"
"while"
"name"
"{"
```

```
### Each of the classes above extends the virtual class
"language"
```

```
"language"
new(Class, ...)Class
```

```
signature(from = "ANY", to = "call")as(object, "call")as.call()
```

```
showClass("language")

is( quote(sin(x)) ) # "call"  "language"

(ff <- new("if")) ; is(ff) # "if"  "language"
(ff <- new("for")) ; is(ff) # "for"  "language"
```

```
LinearMethodsList-class
      "LinearMethodsList"
```

```
"MethodsList"
```

```
linearizeMlist
```

```
methods "list"
arguments "list"
classes "list"
generic "genericFunction"
```

```
linearizeMlistMethodDefinition
```

```
linearizeMlistMethodsList
```

```
LocalReferenceClasses
```

```
setRefClass()contains="localRefClass"
```

```
twiddle
```

```
setRefClass(Class, fields = , contains = c("localRefClass",...),
  methods =, where =, ...)
```

```
$<-
```

```
<<-
```

```
x$ensureLocal()x$field <- ...
```

```

## class "myIter" has a BigData field for the real (big) data
## and a "twiddle" field for some parameters that it twiddles
## ( for some reason)

myIter <- setRefClass("myIter", contains = "localRefClass",
  fields = list(BigData = "numeric", twiddle = "numeric"))

tw <- rnorm(3)
x1 <- myIter(BigData = rnorm(1000), twiddle = tw) # OK, not REALLY big

twiddler <- function(x, n) {
  x$ensureLocal() # see the Details. Not really needed in this example
  for(i in seq_len(n)) {
    x$twiddle <- x$twiddle + rnorm(length(x$twiddle))
    ## then do something ....
    ## Snooping in gdb, etc will show that x$BigData is not copied
  }
  return(x)
}

x2 <- twiddler(x1, 10)

stopifnot(identical(x1$twiddle, tw), !identical(x1$twiddle, x2$twiddle))

```

makeClassRepresentation

[classRepresentationsetClass](#)

```

makeClassRepresentation(name, slots=list(), superClasses=character(),
  prototype=NULL, package, validity, access,
  version, sealed, virtual=NA, where)

```

name	
slots	setClasssuperClasses
superClasses	
prototype	prototype
package	getPackageName
validity	validObject
access	
version	
sealed	setClass
virtual	
where	

setClass

method.skeleton

setMethod

method.skeleton(generic, signature, file, external = FALSE, where)

generic	setMethod
signature	setMethod
file	".R" method.skeleton()
external	setMethod
where	method.skeleton

file

setMethodpackage.skeleton

```
setClass("track", slots = c(x="numeric", y="numeric"))
method.skeleton("show", "track")          ## writes show_track.R
method.skeleton("Ops", c("track", "track")) ## writes "Ops_track_track.R"

## write multiple method skeletons to one file
con <- file("./Math_track.R", "w")
method.skeleton("Math", "track", con)
method.skeleton("exp", "track", con)
method.skeleton("log", "track", con)
close(con)
```

MethodDefinition-class

"function"

targetdefined[setMethod](#)

[setMethod](#)

"SealedMethodDefinition"[setMethod](#)sealed = TRUE"MethodDefinition"

.Data "function"

target "signature"

defined "signature"target

generic "character"

"function"

"PossibleMethod"

"optionalMethod""function"

[MethodsList](#)MethodDefinition[MethodWithNextcallNextMethod](#)

Methods

[setMethod](#)

Methods_Details

[setMethod\(\)](#)[setClass\(\)](#)[setGeneric\(\)](#)[Methods_for_NongenericsClasses_Details](#)

`plot()`

[setMethodGroupGenericFunctions](#)

`package`

`signature`[setMethod](#)"signature"

`expr`[with](#)"missing"

`class(x)`"missing"

`contains=`[setClass](#)[setClassUnion](#)[setIs](#)

[setIs](#)

12[getClass](#)"ANY""ANY""missing""ANY""ANY"

"dgeMatrix""ANY"

`outer`

[testInheritedMethods](#)

[resetGeneric](#)

[selectMethod](#)[coerce\(\)](#)[as\(\)](#)[selectMethod](#)

`missing`

`standardGeneric()`"function"[genericFunction](#)

"generic""package"

`expr`[with](#)

`isS4(x)`TRUE[isGeneric](#)[getGeneric](#)

MethodDefinition"generic""defined""target"setMethod"target""defined""target"
showMethods

setGenericsetMethodsetClass

Methods_for_Nongenerics

setGeneric()

data.frame()as.data.frame()
as.data.frame()data.frame()as.data.frame()

UseMethod()as.data.frame()
chol()qr()svd()
as.data.frame()data.frame()

```

". "UseMethod
f3(x, ...) "myClass"
f3.myClass <- function(x, ...) { ..... }
setMethod("f3", "myClass", f3.myClass)

```

```

setOldClassextends(class(x))
asS3S3PartasS3(x)S3Part

```

```

## A class that extends a registered S3 class inherits that class' S3
## methods.

```

```

setClass("myFrame", contains = "data.frame",
         slots = c(timestamps = "POSIXt"))
df1 <- data.frame(x = 1:10, y = rnorm(10), z = sample(letters,10))
mydf1 <- new("myFrame", df1, timestamps = Sys.time())

```

```

## "myFrame" objects inherit "data.frame" S3 methods; e.g., for `[`

mydf1[1:2, ] # a data frame object (with extra attributes)

```

```

## a method explicitly for "myFrame" class

```

```

setMethod("[",
  signature(x = "myFrame"),
  function (x, i, j, ..., drop = TRUE)
  {
    S3Part(x) <- callNextMethod()
    x@timestamps <- c(Sys.time(), as.POSIXct(x@timestamps))
    x
  }
)

```

```

mydf1[1:2, ]

```

```

setClass("myDateTime", contains = "POSIXt")

```

```

now <- Sys.time() # class(now) is c("POSIXct", "POSIXt")
nowLt <- as.POSIXlt(now) # class(nowLt) is c("POSIXlt", "POSIXt")

```

```

mCt <- new("myDateTime", now)
mLt <- new("myDateTime", nowLt)

```

```

## S3 methods for an S4 object will be selected using S4 inheritance
## Objects mCt and mLt have different S3Class() values, but this is
## not used.
f3 <- function(x) UseMethod("f3") # an S3 generic to illustrate inheritance

f3.POSIXct <- function(x) "The POSIXct result"
f3.POSIXlt <- function(x) "The POSIXlt result"
f3.POSIXt <- function(x) "The POSIXt result"

stopifnot(identical(f3(mCt), f3.POSIXt(mCt)))
stopifnot(identical(f3(mLt), f3.POSIXt(mLt)))

## An S4 object selects S3 methods according to its S4 "inheritance"

setClass("classA", contains = "numeric",
        slots = c(realData = "numeric"))

Math.classA <- function(x) { (getFunction(.Generic))(x@realData) }
setMethod("Math", "classA", Math.classA)

x <- new("classA", log(1:10), realData = 1:10)

stopifnot(identical(abs(x), 1:10))

setClass("classB", contains = "classA")

y <- new("classB", x)

stopifnot(identical(abs(y), abs(x))) # (version 2.9.0 or earlier fails here)

## an S3 generic: just for demonstration purposes
f3 <- function(x, ...) UseMethod("f3")

f3.default <- function(x, ...) "Default f3"

## S3 method (only) for classA
f3.classA <- function(x, ...) "Class classA for f3"

## S3 and S4 method for numeric
f3.numeric <- function(x, ...) "Class numeric for f3"
setMethod("f3", "numeric", f3.numeric)

## The S3 method for classA and the closest inherited S3 method for classB
## are not found.

f3(x); f3(y) # both choose "numeric" method

## to obtain the natural inheritance, set identical S3 and S4 methods
setMethod("f3", "classA", f3.classA)

f3(x); f3(y) # now both choose "classA" method

## Need to define an S3 as well as S4 method to use on an S3 object
## or if called from a package without the S4 generic

```

```

MathFun <- function(x) { # a smarter "data.frame" method for Math group
  for (i in seq_len(ncol(x))[sapply(x, is.numeric)])
    x[, i] <- (getFunction(.Generic))(x[, i])
  x
}
setMethod("Math", "data.frame", MathFun)

## S4 method works for an S4 class containing data.frame,
## but not for data.frame objects (not S4 objects)

try(logIris <- log(iris)) #gets an error from the old method

## Define an S3 method with the same computation

Math.data.frame <- MathFun

logIris <- log(iris)

```

Methods_for_S3

[setOldClass](#)

[UseMethod](#)math

"numeric"

methfunClassNAMESPACE

S3method(fun, Class, meth)

fun.Class".

"uncased""character"[unique](#)

setClass("uncased", contains = "character")

unique.uncased <- function(x, incomparables = FALSE, ...) nextMethod(tolower(x))

setMethod("unique", "uncased", unique.uncased)

NAMESPACE

```
S3method(unique, uncased)
exportMethods(unique)
unique
```

```
unique()
```

```
unique
setOldClassextends(class(x))
```

```
MethodWithNext-class "MethodWithNext"
```

```
callNextMethod
```

```
callNextMethod
```

```
.Data "function"
nextMethod "PossibleMethod"callNextMethod()
excluded "list"
target "signature""MethodDefinition"
defined "signature""MethodDefinition"
generic "character"
```

```
"MethodDefinition"
"function"
"PossibleMethod""MethodDefinition"
"optionalMethod""MethodDefinition"
```

```
signature(method = "MethodWithNext")
signature(object = "MethodWithNext")
```

```
callNextMethodMethodDefinition
```

new

```
newinitialize...initialize()
setClassnew()
```

```
new(Class, ...)
```

```
initialize(.Object, ...)
```

```
Class          charactergetClass
...            initialize()
.Object
```

```
initializenewinitialize()
...new()"initialize"
...
initialize.....callNextMethod
initialize-methods
initializesimpleInheritanceOnlysetGenericsetIs
"numeric"new
```

```
setOldClass
```

```
## using the definition of class "track" from \link{setClass}
```

```
## a new object with two slots specified
t1 <- new("track", x = seq_along(ydata), y = ydata)
```

```
# a new object including an object from a superclass, plus a slot
t2 <- new("trackCurve", t1, smooth = ysmooth)
```

```
### define a method for initialize, to ensure that new objects have
### equal-length x and y slots. In this version, the slots must still be
### supplied by name.
```

```
setMethod("initialize", "track",
  function(.Object, ...) {
```



```

        .Object <- callNextMethod()
        if(length(.Object@x) != length(.Object@y))
            stop("specified x and y of different lengths")
        .Object
    })

### An alternative version that allows x and y to be supplied
### unnamed. A still more friendly version would make the default x
### a vector of the same length as y, and vice versa.

setMethod("initialize", "track",
  function(.Object, x = numeric(0), y = numeric(0), ...) {
    .Object <- callNextMethod(.Object, ...)
    if(length(x) != length(y))
      stop("specified x and y of different lengths")
    .Object@x <- x
    .Object@y <- y
    .Object
  })

```

nonStructure-class

[structurenonStructure](#)

[OpsMath](#)

[OpsMathMath2](#)

[structure](#)

```

setClass("NumericNotStructure", contains = c("numeric","nonStructure"))
xx <- new("NumericNotStructure", 1:10)
xx + 1 # vector
log(xx) # vector
sample(xx) # vector

```

ObjectsWithPackage-class

package

[getGenerics](#)

.Data "character"

package "character"

"character"

"vector""character"

Methods

promptClass

```
promptClass(clName, filename = NULL, type = "class",
            keywords = "classes", where = topenv(parent.frame()),
            generatorName = clName)
```

clName

filename ".Rd"NA

type

keywords "classes"

where

generatorName

[getGenerics](#)

filenameNApromptClass

filenameNACat(unlist(x), file = filename, sep = "\n")x

generatorName

filenameNA

<stvjc@channing.harvard.edu>

[promptpromptMethods](#)

R CMD [Rdconvman](#)

```
## Not run: > promptClass("track")
A shell of class documentation has been written to the
file "track-class.Rd".
```

```
## End(Not run)
```

promptMethods

```
promptMethods(f, filename = NULL, methods)
```

```
f
filename      f"-methods.Rd"FALSENA
methods       "listOfMethods"f
               findMethods(f, where)wheref
```

```
filenameFALSEprompt
filenameNAcat(unlist(x), file = filename, sep = "\n")x
filename
```

```
filenameFALSEfilenameNA
```

[promptpromptClass](#)

ReferenceClasses

`\$`

setRefClass\$methods()

contains=setRefClass

setRefClass(Class, fields = , contains = , methods =,
where =, inheritPackage =, ...)

getRefClass(Class, where =)

Class

getRefClass()

fields

contains

methods \$methods

where setRefClass

getRefClasses[asNamespace](#)

inheritPackage FALSE

... [setClass](#)

setRefClass()initialize

[setClass](#)

getRefClass()where

"\$"m1xx\$m1(...)

"ANY"

.self

```

$methods()gmethodssetRefClass

.self.self.self$x
<<-$edit$undo<-

initializefinalizeinitialize$new(...)
finalizeatexit=TRUE

.self.refClassDef.self$initialize

$usingMethods()sapply\(\)do.call\(\)
$helpRd$help

$initialize()initialize()$initialize()$new()$new()
$initialize()$initFields(...)
missing()...$callSuper()$initFields()

contains=refSuperClasses

callSuper(...)

"envRefClass"

$callSuper(...) $callSuper

$copy(shallow = FALSE) shallowFALSEshallow = TRUE
$field(name, value) namevaluename
    $field()
$export(Class) Class
$getRefClass()$getClass()
$import(value, Class = class(value)) valuevalueClassvalue
$initFields(...) $initialize()
    $initialize()$initFields()matrixViewer
$show() show"envRefClass"
    show()show()$show()methods::show()
$trace(what, ...)$untrace(what) tracewhat
    tracesignature

```

```
$usingMethods(...) $usingMethods()
```

```
.self
```

```
.refClassDef
```

```
". "
```

```
"$"
```

```
inheritPackagesetRefClass()
```

```
.self
```

```
.self$name
```

```
.selfff()externalRefMethod(f)"externalRefMethod"
```

```
setRefClass"refObjectGenerator""function""classGeneratorFunction"
```

```
"refGeneratorSlot""refGeneratorSlot""refObjectGenerator"
```

```
defclassName
```

```
$new(...) setRefClass
```

```
$help(topic)
```

```
$methods(...)
```

```
    methodssetRefClass()setRefClass()
```

```
    $methods()
```

```
    NULL
```

```
$fields() "activeBindingFunction"
```

```
$lock(...)
```

```
$trace(what, ..., classMethod = FALSE) what$trace()
```

```
    classMethod = TRUEwhat
```

```
$accessors(...) abcx$getAbc()x$setAbc(value)$accessors
```

```

"environment"
fields=setRefClass
datax$datadatax$data <- valutex <<- value

```

```

setValidityvalidObjectobject$

```

```

debugdebug(xx$edit)
trace$trace()xx$trace()mEdit$trace()$trace()$trace()
tracesignature=trace(what, browser)trace(what, edit = TRUE)

```

```

## a simple editor for matrix objects. Method $edit() changes some
## range of values; method $undo() undoes the last edit.

```

```

mEdit <- setRefClass("mEdit",
  fields = list( data = "matrix",
    edits = "list"))

```

```

## The basic edit, undo methods

```

```

mEdit$methods(
  edit = function(i, j, value) {
    ## the following string documents the edit method
    'Replaces the range [i, j] of the
    object by value.
    ',
    backup <-
      list(i, j, data[i,j])
    data[i,j] <<- value
    edits <<- c(edits, list(backup))
    invisible(value)
  },
  undo = function() {
    'Undoes the last edit() operation
    and update the edits field accordingly.
    ',
    prev <- edits
    if(length(prev)) prev <- prev[[length(prev)]]
    else stop("No more edits to undo")
    edit(prev[[1]], prev[[2]], prev[[3]])
    ## trim the edits list
    length(edits) <<- length(edits) - 2
    invisible(prev)
  })

```

```

## A method to automatically print objects

```

```

mEdit$methods(
  show = function() {
    'Method for automatically printing matrix editors'
    cat("Reference matrix editor object of class",
        classLabel(class(.self)), "\n")
    cat("Data: \n")
    methods::show(data)
    cat("Undo list is of length", length(edits), "\n")
  }
)

xMat <- matrix(1:12,4,3)
xx <- mEdit(data = xMat)
xx$edit(2, 2, 0)
xx
xx$undo()
mEdit$help("undo")
stopifnot(all.equal(xx$data, xMat))

utils::str(xx) # show fields and names of methods

## A method to save the object
mEdit$methods(
  save = function(file) {
    'Save the current object on the file
    in R external object format.
    '
    base::save(.self, file = file)
  }
)

tf <- tempfile()
xx$save(tf)

## Not run:
## Inheriting a reference class: a matrix viewer
mv <- setRefClass("matrixViewer",
  fields = c("viewerDevice", "viewerFile"),
  contains = "mEdit",
  methods = list( view = function() {
    dd <- dev.cur(); dev.set(viewerDevice)
    devAskNewPage(FALSE)
    matplot(data, main = paste("After",length(edits),"edits"))
    dev.set(dd)},
    edit = # invoke previous method, then replot
    function(i, j, value) {
      callSuper(i, j, value)
      view()
    })
)

## initialize and finalize methods
mv$methods( initialize =
  function(file = "../matrixView.pdf", ...) {
    viewerFile <- file
    pdf(viewerFile)
    viewerDevice <- dev.cur()
  }
)

```



```

        dev.set(dev.prev())
        callSuper(...)
    },
    finalize = function() {
        dev.off(viewerDevice)
    })

## debugging an object: call browser() in method $edit()
xx$trace(edit, browser)

## debugging all objects from class mEdit in method $undo()
mEdit$trace(undo, browser)

## End(Not run)

```

removeMethod

```
removeMethod(f, signature, where)
```

```

fsignaturewhere
    setMethod()

```

```
TRUE
```

representation

```

representation()slotscontainssetClass
prototype()

```

```

representation(...)
prototype(...)

```

```

...
    prototype

```

```
representation
representation
prototype
```

```
representation
prototype
```

setClass

```
## representation for a new class with a directly define slot "smooth"
## which should be a "numeric" object, and extending class "track"
representation("track", smooth ="numeric")
```

```
### >>> This *is* old syntax -- use 'contains=*, slots=*' instead <<<
###          =====          -----          =====
```

```
setClass("Character",representation("character"))
setClass("TypedCharacter",representation("Character",type="character"),
         prototype(character(0),type="plain"))
ttt <- new("TypedCharacter", "foo", type = "character")
```

```
setClass("num1", representation(comment = "character"),
         contains = "numeric",
         prototype = prototype(pi, comment = "Start with pi"))
```

S3Part

setOldClass

```
as(object, S3Class); as(object, "S3")
S3Class
```

```

S3Part(object, strictS3 = FALSE, S3Class)

S3Class(object)

isXS3Class(classDef)

slotsFromS3(object)

## the replacement versions of the functions are not recommended
## Create a new object from the class or use the replacement version of as().

S3Part(object, strictS3 = FALSE, needClass = ) <- value

S3Class(object) <- value


object

strictS3      TRUE S3Part setOldClass
S3Class       character object
classDef       getClass

needClass

value         S3Part <-
              S3Class <- class(x)


S3Partcontains= setClass
strictS3 = TRUE S3Part() strictS3 = FALSE S3Part()
"S3" as()
S3Class.S3Class class
isXS3Class TRUE FALSE classDef
slotsFromS3
slotsFromS3() setOldClass S4Class S4Class setOldClass


class() isS4() TRUE FALSE as(x, "S3")
"data.frame" "lm" "list" "lm"
setOldClass ".S3Class" "lm" "mlm" "xlm"
"ts" "matrix" "array" as(x, "matrix") as(x, "S3") S3Part(x)

```

[isS4asS4](#)

```
"S3""S4"as(object, "S3")as(object, "S4")
as(object, "S3")class(object)
as(object, "S3")
as(object, "S4")class(object)new
```

[setOldClass](#)

```
## an "mlm" object, regressing two variables on two others

sepal <- as.matrix(datasets::iris[,c("Sepal.Width", "Sepal.Length")])
fit <- lm(sepal ~ Petal.Length + Petal.Width + Species, data = datasets::iris)
class(fit) # S3 class: "mlm", "lm"

## a class that contains "mlm"
myReg <- setClass("myReg", slots = c(title = "character"), contains = "mlm")

fit2 <- myReg(fit, title = "Sepal Regression for iris data")

fit2 # shows the inherited "mlm" object and the title

identical(S3Part(fit2), as(fit2, "mlm"))

class(as(fit2, "mlm")) # the S4 class, "mlm"

class(as(fit2, "S3")) # the S3 class, c("mlm", "lm")

## An object may contain an S3 class from a subclass of that declared:
xlm <- setClass("xlm", slots = c(eps = "numeric"), contains = "lm")

xfit <- xlm(fit, eps = .Machine$double.eps)

xfit@.S3Class # c("mlm", "lm")
```

S4groupGeneric

[setGroupGeneric](#)

```
## S4 group generics:
Arith(e1, e2)
Compare(e1, e2)
Ops(e1, e2)
matrixOps(x, y)
Logic(e1, e2)
Math(x)
Math2(x, digits)
Summary(x, ..., na.rm = FALSE)
Complex(z)
```

```
xyze1e2
digits          roundsignif
...
na.rm
```

[setMethodMethods_Details](#)

MathOpsSummaryComplexmatrixOps?[S3groupGeneric](#)

[getGroupMembers](#)

```
Arith  "+" "-" "*" "^" "%%" "%/%" "/"
Compare  "==" ">" "<" "!=" "<=" ">="
Logic  "&" "|"
Ops  "Arith" "Compare" "Logic"
Math  "abs" "sign" "sqrt" "ceiling" "floor" "trunc" "cummax" "cummin" "cumprod" "cumsum"
      "log" "log10" "log2" "log1p" "acos" "acosh" "asin" "asinh" "atan" "atanh" "exp"
      "expm1" "cos" "cosh" "cospi" "sin" "sinh" "sinpi" "tan" "tanh" "tanpi" "gamma"
      "lgamma" "digamma" "trigamma"
Math2  "round" "signif"
Summary  "max" "min" "range" "prod" "sum" "any" "all"
Complex  "Arg" "Conj" "Im" "Mod" "Re"
matrixOps  "%*%"
```

Ops

Math[logtrunc](#)Math

[callGeneric](#)

```
setClass("testComplex", slots = c(zz = "complex"))
## method for whole group "Complex"
getGroupMembers("Complex") # "Arg" "Conj" "Im" "Mod" "Re"
setMethod("Complex", "testComplex",
  function(z) c("groupMethod", callGeneric(z@zz)))
## exception for Arg() :
setMethod("Arg", "testComplex",
  function(z) c("ArgMethod", Arg(z@zz)))
z1 <- 1+2i
z2 <- new("testComplex", zz = z1)
stopifnot(identical(Mod(z2), c("groupMethod", Mod(z1))))
stopifnot(identical(Arg(z2), c("ArgMethod", Arg(z1))))
selectMethod("Re", signature = "testComplex") # shows Generic: .. "Re" & .."Complex"
```

SClassExtension-class

[setIscontains=setClasssetClassUnion](#)

```
subClasssuperClass fromtosetIs
package
coerce strict=TRUEEas
test testsetIs
replace as(x, Class) <- value
simple "logical"TRUE
by
dataPart "logical"TRUE
distance
```

"SClassExtension"

[isasclassRepresentation](#)

`selectSuperClasses`

`ClassDef`
`contains`

```
selectSuperClasses(Class, dropVirtual = FALSE, namesOnly = TRUE,  
                    directOnly = TRUE, simpleOnly = directOnly,  
                    where = toplevel(parent.frame()))  
  
.selectSuperClasses(ext, dropVirtual = FALSE, namesOnly = TRUE,  
                    directOnly = TRUE, simpleOnly = directOnly)
```

```
Class          getClass  
dropVirtual  
namesOnly  
directOnly  
simpleOnly  
where          ClassClass  
ext            .selectSuperClasses()listgetClassDef(..)@contains
```

`character``namesOnly``contains``getClass`

`selectSuperClasses().selectSuperClasses()`

`isgetClass``classRepresentation`

```
setClass("Root")  
setClass("Base", contains = "Root", slots = c(length = "integer"))  
setClass("A", contains = "Base", slots = c(x = "numeric"))  
setClass("B", contains = "Base", slots = c(y = "character"))  
setClass("C", contains = c("A", "B"))  
  
extends("C") #--> "C" "A" "B" "Base" "Root"  
selectSuperClasses("C") # "A" "B"  
selectSuperClasses("C", directOnly=FALSE) # "A" "B" "Base" "Root"  
selectSuperClasses("C", dropVirtual=TRUE, directOnly=FALSE)# ditto w/o "Root"
```

setAs

```
setAsfromtoasfrom  
coercecoerce
```

```
setAs(from, to, def, replace, where = toenv(parent.frame()))
```

```
fromto      defreplace  
def         fromtofromsetAsfrom  
replace     asfrom, valuesetAs
```

```
where
```

```
ascontains=setClass  
setClasssetIsas  
setAs  
setIsfromsetAs  
setIs
```

```
assetAs
```

```
asobjectClasscoerce`coerce<-`fromtoas`coerce<-`replacesetAscoercedef  
objectascoercec(from = class(object), to = Class)  
asClassclass(object)setIs  
asfromtoClassselectMethod("coerce", sig, useInherited= c(from=TRUE, to= FALSE))  
as()  
class(object)Class  
coerceasselectMethodasas  
setAscoercec(from, to)defdeffromtoasetAstoasfromtoto  
asstrict=TRUEFALSEtostrict=FALSEstrict=  
replacesetAs`coerce<-`valuecoercefrom, tostrict=  
coerceasstandardGenerictoascoerceas
```

```
as(x, "numeric")as.numericshowMethods("coerce")
```

```
try(as(x, cl))canCoerce(x, cl)
```



```
## using the definition of class "track" from \link{setClass}

setAs("track", "numeric", function(from) from@y)

t1 <- new("track", x=1:20, y=(1:20)^2)

as(t1, "numeric")

## The next example shows:
## 1. A virtual class to define setAs for several classes at once.
## 2. as() using inherited information

setClass("ca", slots = c(a = "character", id = "numeric"))

setClass("cb", slots = c(b = "character", id = "numeric"))

setClass("id")
setIs("ca", "id")
setIs("cb", "id")

setAs("id", "numeric", function(from) from@id)

CA <- new("ca", a = "A", id = 1)
CB <- new("cb", b = "B", id = 2)

setAs("cb", "ca", function(from, to )new(to, a=from@b, id = from@id))

as(CB, "numeric")
```

setClass

```
myClass <- setClass("myClass", slots= ..., contains =...)
slots=contains=setClass()

setClass(Class, representation, prototype, contains=character(),
        validity, access, where, version, sealed, package,
        S3methods = FALSE, slots)
```

```
Class
slots          slots=
               getClass
               "ANY"
```

```

contains      contains=
              "VIRTUAL"
prototypewherevaliditysealedpackage

              prototypeinitialize()
              wheresetClass()
              validitysetValidity()
              sealedTRUEsetClass
              package
representationaccessversionS3methods

              representationslotscontains
              accessversion
              S3methods

```

```

newinitialize
new
new()

```

```

slotscontains

```

```

slotsa"ANY"a"numeric""character"setClass
"class""Class""."

```

```

setClassUnion()setClass()
setClass()Classcontains="VIRTUAL"
traceable"VIRTUAL"

```

```

"matrix""array""Data"
typeof(x).Data"numWithId"
"vector""matrix""array"
typeof(x)"S4"
"environment""externalptr""name""xData""stampedEnv""Data"
as.()as.environment(e1)"stampedEnv"typeof(e1)"S4"

```

```

setOldClass
inheritssetOldClasssetOldClass

```

classRepresentation

setClassDESCRIPTIONNAMESPACE"CMD check"

packageSlotname

Classes_DetailsMethods_DetailsmakeClassRepresentation

```
## A simple class with two slots
track <- setClass("track", slots = c(x="numeric", y="numeric"))
## an object from the class
t1 <- track(x = 1:10, y = 1:10 + rnorm(10))

## A class extending the previous, adding one more slot
trackCurve <- setClass("trackCurve",
  slots = c(smooth = "numeric"),
  contains = "track")

## an object containing a superclass object
t1s <- trackCurve(t1, smooth = 1:10)

## A class similar to "trackCurve", but with different structure
## allowing matrices for the "y" and "smooth" slots
setClass("trackMultiCurve",
  slots = c(x="numeric", y="matrix", smooth="matrix"),
  prototype = list(x=numeric(), y=matrix(0,0,0),
    smooth= matrix(0,0,0)))

## A class that extends the built-in data type "numeric"

numWithId <- setClass("numWithId", slots = c(id = "character"),
  contains = "numeric")

numWithId(1:3, id = "An Example")

## inherit from reference object of type "environment"
stampedEnv <- setClass("stampedEnv", contains = "environment",
  slots = c(update = "POSIXct"))
setMethod("[[<-", c("stampedEnv", "character", "missing"),
  function(x, i, j, ..., value) {
    ev <- as(x, "environment")
    ev[[i]] <- value #update the object in the environment
    x@update <- Sys.time() # and the update time
  })

e1 <- stampedEnv(update = Sys.time())

e1[["noise"]] <- rnorm(10)
```

setClassUnion

```
setClassUnion(name, members, where)
isClassUnion(Class)
```

name

members

where

Class

members

`setIs`"numeric"

"maybeNumber""numeric"

"ClassUnionRepresentation"`classRepresentation`

```
## a class for either numeric or logical data
setClassUnion("maybeNumber", c("numeric", "logical"))
```

```
## use the union as the data part of another class
setClass("withId", contains = "maybeNumber", slots = c(id = "character"))
```

```
w1 <- new("withId", 1:10, id = "test 1")
w2 <- new("withId", sqrt(w1)%%1 < .01, id = "Perfect squares")
```

```
## add class "complex" to the union "maybeNumber"
setIs("complex", "maybeNumber")
```

```
w3 <- new("withId", complex(real = 1:10, imaginary = sqrt(1:10)))
```

```
## a class union containing the existing class union "OptionalFunction"
setClassUnion("maybeCode",
  c("expression", "language", "OptionalFunction"))
```

```
is(quote(sqrt(1:10)), "maybeCode") ## TRUE
```

setGeneric

[setMethod](#) [setGeneric](#)

[setGeneric](#)

[setGeneric\(name\)](#)

name

[setGeneric\(name, def\)](#)

def

[setGeneric\(name, def= , group=list\(\), valueClass=character\(\),
where= , package= , signature= , useAsDefault= ,
genericFunction= , simpleInheritanceOnly = \)](#)

name

def [defsetGeneric](#)

group

valueClass

signature [setMethod...](#)

.....

[simpleInheritanceOnly](#)

TRUEcontains=[setClassinitializeshow](#)

useAsDefault

package

where

[genericFunction](#)

[setGenericname](#)

setGeneric

name

setGeneric("colSums")

"base"importFrom()NAMESPACE"base"

"base"

UseMethod"genericFunction"

setGeneric

setGeneric()

setGeneric()setMethod()setMethodsetMethodsetGenericsetMethod

setGeneric()asRObject()setGeneric()

def

setGeneric()namedef

implicitGeneric"methods"implicitGeneric

standardGenericstandardGeneric"authorNames"

defuseAsDefault

valueClassis(object, Class)

defstandardGeneric()

groupsetGroupGeneric

implicitGeneric

setGeneric()unlistas.vector

getGeneric(name)isGeneric

%*%

Methods_DetailsdotsMethods...setMethod

```

## Specify that this package will define methods for plot()
setGeneric("plot")

## create a new generic function, with a default method
setGeneric("props", function(object) attributes(object))

### A non-standard generic function. It insists that the methods
### return a non-empty character vector (a stronger requirement than
### valueClass = "character" in the call to setGeneric)

setGeneric("authorNames",
  function(text) {
    value <- standardGeneric("authorNames")
    if(!(is(value, "character") && any(nchar(value)>0)))
      stop("authorNames methods must return non-empty strings")
    value
  })

## the asRObject generic function, from package XR
## Its default method just returns object
## See the reference, Chapter 12 for methods

setGeneric("asRObject", function(object, evaluator) {
  object
})

```

setGroupGeneric

setGroupGeneric[setGeneric](#)

```

setGroupGeneric(name, def= , group=list(), valueClass=character(),
  knownMembers=list(), package= , where= )

```

name

def

groupvalueClass

[setGeneric](#)

knownMembers

packagewhere [setGeneric](#)

setGroupGenericname

[Methods_DetailsdotsMethods...setMethod](#)

```
## Not run:
## the definition of the "Logic" group generic in the methods package
setGroupGeneric("Logic", function(e1, e2) NULL,
  knownMembers = c("&", "|"))

## End(Not run)
```

setIs

```
setIscontains=setClass
setAs()

setIs(class1, class2, test=NULL, coerce=NULL, replace=NULL,
  by = character(), where = topenv(parent.frame()), classDef =,
  extensionObject = NULL, doComplete = TRUE)

class1class2    is
coercereplace   class2is(object, class2)value
test

extensionObject
      test, coerce, replace, bySClassExtension
doComplete     TRUE
by              setIs
where          setIs
classDef       classsetIssetClass

      contains=setClasssimple
      class1setClassUnionsetIs
      coercereplacesetAssetAssetAsassetIs

setAs

coerce=replace=setIs
coercereplaceclass1class2class2fromfromvalue
byclass2by
setIsclass1
```



```

coerceclass1class2replaceclass1class2valueas(object, class2) <- value
coercereplaceclass1class2setIs
coerceclass2class1replaceclass1class2
setAsdefcoerce
testTRUEFALSEis(object, class2)

```

```

contains=setClass
setIsas
asstrict = FALSE
class2class1class2class1"factor""character""factor"
initializesimpleInheritanceOnlysetGeneric
setIssetAsclass1

```

```

## Two examples of setIs() with coerce= and replace= arguments
## The first one works fairly well, because neither class has many
## inherited methods do be disturbed by the new inheritance

## The second example does NOT work well, because the new superclass,
## "factor", causes methods to be inherited that should not be.

## First example (classes taken from examples in ?setClass):
## A simple class with two slots
setClass("track", slots = c(x="numeric", y="numeric"))
## A class extending the previous, adding one more slot
setClass("trackCurve", contains = "track",
        slots = c(smooth = "numeric"))
## A class similar to "trackCurve", but with different structure
## allowing matrices for the "y" and "smooth" slots
setClass("trackMultiCurve",
        slots = c(x="numeric", y="matrix", smooth="matrix"),
        prototype = structure(list(), x=numeric(), y=matrix(0,0,0),
                               smooth=matrix(0,0,0)))

## Automatically convert an object from class "trackCurve" into
## "trackMultiCurve", by making the y, smooth slots into 1-column matrices
setIs("trackCurve",
      "trackMultiCurve",
      coerce = function(obj) {
        new("trackMultiCurve",
            x = obj@x,
            y = as.matrix(obj@y),
            smooth = as.matrix(obj@smooth))
      },

```

```

replace = function(obj, value) {
  obj@y <- as.matrix(value@y)
  obj@x <- value@x
  obj@smooth <- as.matrix(value@smooth)
  obj}}

## Second Example:
## A class that adds a slot to "character"
setClass("stringsDated", contains = "character",
  slots = c(stamp="POSIXt"))

## Convert automatically to a factor by explicit coerce
setIs("stringsDated", "factor",
  coerce = function(from) factor(from@.Data),
  replace= function(from, value) {
    from@.Data <- as.character(value); from })

l1 <- sample(letters, 10, replace = TRUE)
ld <- new("stringsDated", l1, stamp = Sys.time())

levels(as(ld, "factor"))
levels(ld) # will be NULL--see comment in section on inheritance above.

## In contrast, a class that simply extends "factor"
## has no such ambiguities
setClass("factorDated", contains = "factor",
  slots = c(stamp="POSIXt"))
fd <- new("factorDated", factor(l1), stamp = Sys.time())
identical(levels(fd), levels(as(fd, "factor")))

```

setLoadActions

```

setLoadAction()setLoadActions()
getLoadActions
hasLoadActionTRUEwhere
evalOnLoad()evalqOnLoad()

setLoadAction(action, aname=, where=)

setLoadActions(..., .where=)

getLoadActions(where=)

hasLoadAction(aname, where=)

evalOnLoad(expr, where=, aname=)

evalqOnLoad(expr, where=, aname=)

```

action...

where.where

aname ".1"

expr whereevalqOnLoad()evalOnLoad()"language"

evalOnLoad()evalqOnLoad()setLoadAction()

setLoadAction()setLoadActions()setLoadActions()setLoadActions()".1"

setLoadAction()setLoadActions()setLoadAction()setLoadActions()setLoadAction()

.onLoad()

setHook()

setLoadAction()setLoadActions()

getLoadActions()

hasLoadAction()TRUE

setHook

Not run:

in the code for some package

... somewhere else

setLoadActions(function(ns)

cat("Loaded package", sQuote(getNamespaceName(ns)),

"at", format(Sys.time()), "\n"),

setCount = function(ns) assign("myCount", 1, envir = ns),

function(ns) assign("myPointer", getMyExternalPointer(), envir = ns))

... somewhere later

if(countShouldBe0)

setLoadAction(function(ns) assign("myCount", 0, envir = ns), "setCount")

End(Not run)

setMethod

```
setMethod(f, signature, definition)
fsignaturedefinition
```

```
setMethod(f, signature=character(), definition,
          where = toplevel(parent.frame()),
          valueClass = NULL, sealed = FALSE)
```

```
f
signature      method.skeletonsetMethod
definition      fsignaturesetMethod()...
wherevalueClasssealed

          where
          valueClass
          sealed
```

```
xsetMethodx
class(x)contains=
class(x)
```

[ArithOps](#)

```
exportMethods()NAMESPACE
plot()
```

```
setMethod
"show""methods""base"setGeneric
```

```
"numeric""character""matrix""ANY""missing""ANY""missing"setOldClass
standardGeneric(f)"missing"
setMethodf
f"ANY"
```

```
method.skeleton setMethod
dotsMethods setGeneric
```

```
## examples for a simple class with two numeric slots.
## (Run example(setMethod) to see the class and function definitions)

## methods for plotting track objects
##
## First, with only one object as argument, plot the two slots
## y must be included in the signature, it would default to "ANY"
setMethod("plot", signature(x="track", y="missing"),
  function(x, y, ...) plot(x@x, x@y, ...)
)

## plot numeric data on either axis against a track object
## (reducing the track object to the cumulative distance along the track)
## Using a short form for the signature, which matches like formal arguments
setMethod("plot", c("track", "numeric"),
  function(x, y, ...) plot(cumdist(x@x, x@y), y, xlab = "Distance",...)
)

## and similarly for the other axis
setMethod("plot", c("numeric", "track"),
  function(x, y, ...) plot(x, cumdist(y@x, y@y), ylab = "Distance",...)
)

t1 <- new("track", x=1:20, y=(1:20)^2)
plot(t1)
plot(qnorm(ppoints(20)), t1)

## Now a class that inherits from "track", with a vector for data at
## the points
setClass("trackData", contains = c("numeric", "track"))

tc1 <- new("trackData", t1, rnorm(20))

## a method for plotting the object
## This method has an extra argument, allowed because ... is an
## argument to the generic function.
setMethod("plot", c("trackData", "missing"),
  function(x, y, maxRadius = max(par("cin")), ...) {
    plot(x@x, x@y, type = "n", ...)
    symbols(x@x, x@y, circles = abs(x), inches = maxRadius)
  }
)
plot(tc1)
```

```

## Without other methods for "trackData", methods for "track"
## will be selected by inheritance

plot(qnorm(ppoints(20)), tc1)

## defining methods for primitive function.
## Although "[" and "length" are not ordinary functions
## methods can be defined for them.
setMethod("[" , "track",
  function(x, i, j, ..., drop) {
    x@x <- x@x[i]; x@y <- x@y[i]
    x
  })
plot(t1[1:15])

setMethod("length", "track", function(x)length(x@y))
length(t1)

## Methods for binary operators
## A method for the group generic "Ops" will apply to all operators
## unless a method for a more specific operator has been defined.

## For one trackData argument, go on with just the data part
setMethod("Ops", signature(e1 = "trackData"),
  function(e1, e2) callGeneric(e1@.Data, e2))

setMethod("Ops", signature(e2 = "trackData"),
  function(e1, e2) callGeneric(e1, e2@.Data))

## At this point, the choice of a method for a call with BOTH
## arguments from "trackData" is ambiguous. We must define a method.

setMethod("Ops", signature(e1 = "trackData", e2 = "trackData"),
  function(e1, e2) callGeneric(e1@.Data, e2@.Data))
## (well, really we should only do this if the "track" part
## of the two arguments matched)

tc1 +1

1/tc1

all(tc1 == tc1)

```

```
setOldClass
```

```

setOldClass(Classes)
Classes$classsetOldClass()
S4Class=setOldClass()

```

```
setOldClass(Classes, prototype, where, test = FALSE, S4Class)
```

```
Classes      class
```

```
S4Class      Classes  
prototype    wheretest
```

```
      prototypeVIRTUALS4Class=  
      where  
      testTRUE
```

```
ClassesoldClass"mlm"  
NULL
```

```
setOldClass test=TRUE
```

```
.OldClassesListsetOldClasssetMethod
```

```
setOldClass()S4Class=setOldClass()  
"ts"data.framesetClasssetOldClass  
"ts""tsp""ts""tsp"  
"ts"  
dimnames"lm""NULL"NULL  
"terms""data.frame"model.framevalidObject
```

```
setClasssetMethod
```

```
require(stats)
```

```
## "lm" and "mlm" are predefined; if they were not this would do it:  
## Not run:  
setOldClass(c("mlm", "lm"))  
## End(Not run)
```

```
## Define a new generic function to compute the residual degrees of freedom
setGeneric("dfResidual",
  function(model) stop(gettextf(
    "This function only works for fitted model objects, not class %s",
    class(model))))

setMethod("dfResidual", "lm", function(model)model$df.residual)

## dfResidual will work on mlm objects as well as lm objects
myData <- data.frame(time = 1:10, y = (1:10)^.5)
myLm <- lm(cbind(y, y^3) ~ time, myData)

## two examples extending S3 class "lm": class "xlm" directly
## and "ylm" indirectly
setClass("xlm", slots = c(eps = "numeric"), contains = "lm")
setClass("ylm", slots = c(header = "character"), contains = "xlm")
ym1 = new("ylm", myLm, header = "Example", eps = 0.)
## for more examples, see ?\link{S3Class}.

## Not run:
## The code in R that defines "ts" as an S4 class
setClass("ts", contains = "structure", slots = c(tsp = "numeric"),
  prototype(NA, tsp = rep(1,3)))
  # prototype to be a legal S3 time-series
## and now registers it as an S3 class
setOldClass("ts", S4Class = "ts", where = envir)

## End(Not run)
```

show

[showDefault](#)

show

show(object)

object

[setClass](#)[showprint](#)

showsimpleInheritanceOnly[setGeneric](#)[setIs](#)

showNULL

showMethods

```
## following the example shown in the setMethod documentation ...
setClass("track", slots = c(x="numeric", y="numeric"))
setClass("trackCurve", contains = "track", slots = c(smooth = "numeric"))

t1 <- new("track", x=1:20, y=(1:20)^2)

tc1 <- new("trackCurve", t1)

setMethod("show", "track",
  function(object)print(rbind(x = object@x, y=object@y))
)
## The method will now be used for automatic printing of t1

t1

## Not run:   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x   1   2   3   4   5   6   7   8   9  10  11  12
y   1   4   9  16  25  36  49  64  81 100 121 144
  [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
x   13  14  15  16  17  18  19  20
y  169 196 225 256 289 324 361 400

## End(Not run)
## and also for tc1, an object of a class that extends "track"
tc1

## Not run:   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
x   1   2   3   4   5   6   7   8   9  10  11  12
y   1   4   9  16  25  36  49  64  81 100 121 144
  [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
x   13  14  15  16  17  18  19  20
y  169 196 225 256 289 324 361 400

## End(Not run)
```

showMethods

```
showMethods(f = character(), where = topenv(parent.frame()),
  classes = NULL, includeDefs = FALSE,
  inherited = !includeDefs,
  showEmpty, printTo = stdout(), fdef)
.S4methods(generic.function, class)
```

```
f
      fdefisDiagonal()
where      fwheregetGenerics(where)where
classes    classes
includeDefs includeDefsTRUE
inherited   selectMethod
showEmpty   TRUEf
printTo
fdef        where
generic.functionclass
              methods
```

```
methods.S4methods
```

```
fshowMethods fdef
```

```
printToFALSEinvisible
```

```
setMethodGenericFunctionsselectMethod
methods
```

```
require(graphics)
```

```
## Assuming the methods for plot
## are set up as in the example of help(setMethod),
## print (without definitions) the methods that involve class "track":
showMethods("plot", classes = "track")
## Not run:
# Function "plot":
# x = ANY, y = track
# x = track, y = missing
# x = track, y = ANY
```

```
require("Matrix")
showMethods("%*%")# many!
  methods(class = "Matrix")# nothing
showMethods(class = "Matrix")# everything
showMethods(Matrix:::isDiagonal) # a non-exported generic
```

```
## End(Not run)
```

```

if(no4 <- is.na(match("stats4", loadedNamespaces()))
  loadNamespace("stats4")
showMethods(classes = "mle") # -> a method for show()
if(no4) unloadNamespace("stats4")

```

signature-class	"signature"
-----------------	-------------

MethodDefinition

```
new("signature", functionDef, ...)functionDefpackage
```

```

.Data "character"
names "character"
package "character"package

```

```
"character""vector"
```

```
signature(object = "signature")
```

MethodDefinition

slot

```

object@name
object@name <- value

slot(object, name)
slot(object, name, check = TRUE) <- value
.hasSlot(object, name)

slotNames(x)
.slotNames(x)
getSlots(x)

```

```

object
name      .
          slotname

value
check     slotTRUEFALSE
x         slotNamesclass(x)

```

```

@slotslotcheck=FALSE
check=FALSEattr<-
"@
"@slot()slot()as(value, slotClass, strict = FALSE)"@
"@as()slot()

"@slot
slotNamesgetSlotsxslotNames(x)names(getSlots(x))

```

[@Classes_DetailsMethods_DetailsgetClassnames](#)

```

setClass("track", slots = c(x="numeric", y="numeric"))
myTrack <- new("track", x = -4:4, y = exp(-4:4))
slot(myTrack, "x")
slot(myTrack, "y") <- log(slot(myTrack, "y"))
utils::str(myTrack)

getSlots("track") # or
getSlots(getClass("track"))
slotNames(class(myTrack)) # is the same as
slotNames(myTrack)

## Transform such an S4 object to a list, e.g. to "export" it:
S4toList <- function(obj) {
  sn <- slotNames(obj)
  structure(lapply(sn, slot, object = obj), names = sn)
}
S4toList(myTrack)

```

StructureClasses

```
structure
```

```
## The following class names can appear in method signatures,  
## as the class in as() and is() expressions, and, except for  
## the classes commented as VIRTUAL, in calls to new()
```

```
"matrix"  
"array"  
"ts"
```

```
"structure" ## VIRTUAL
```

```
new(Class, ...)Class"matrix"matrix()  
"matrix""array""class"is.objectisS4FALSE  
contains = "matrix"setGeneric"matrix"  
"matrix""array".Data.DataClasses_Details  
"ts"setOldClass"tsp""mts"  
"ts"setOldClass"mts""matrix""array"  
initializematrixarraytsnewinitialize
```

```
"structure""vector""structure"
```

```
as(x, "matrix")as.matrix(x)strict = TRUEas()dimdimnames  
S4groupGeneric  
"vector"
```

```
showClass("structure")
```

```
## explore a bit :  
showClass("ts")  
(ts0 <- new("ts"))  
str(ts0)
```

```
showMethods("Ops") # six methods from these classes, but maybe many more
```

testInheritedMethods

```
testInheritedMethods(f, signatures, test = TRUE, virtual = FALSE,
                     groupMethods = TRUE, where = .GlobalEnv)
```

f

signatures testInheritedMethodstest = FALSE

test FALSEselectMethod

virtual

groupMethods

where

findMethodSignatures"vector""ANY""Other"

selectMethodtestInheritedMethods"ambiguousMethodSelection"

"methodSelectionReport""target""selected""candidates""note""generic"
"allSelections"

<https://johnmchambers.su.domains/classInheritance.pdf>

```
## if no other attached packages have methods for `+` or its group
## generic functions, this returns a 16 by 2 matrix of selection
## patterns (in R 2.9.0)
testInheritedMethods("+")
```

TraceClasses

`traceuntrace`

```
### Objects from the following classes are generated
### by calling trace() on an object from the corresponding
### class without the "WithTrace" in the name.
```

```
"functionWithTrace"
"MethodDefinitionWithTrace"
"MethodWithNextWithTrace"
"genericFunctionWithTrace"
"groupGenericFunctionWithTrace"
```

```
### the following is a virtual class extended by each of the
### classes above
```

```
"traceable"
```

```
traceinitialize"traceable"
```

```
.Data "function""functionWithTrace"
original "function""functionWithTrace"
```

```
"functionWithTrace""function""traceable""VIRTUAL""traceable"
```

```
trace()
```

`trace`

validObject

```
validObject()object
```

```
TRUEtestTRUE
```

```
initialize()validObject
```

```
setValidityvaliditysetClassTRUEvalidObjectcontains=setClass
```

```
validObject(object, test = FALSE, complete = FALSE)

setValidity(Class, method, where = topenv(parent.frame())) )

getValidity(ClassDef)
```

```
object
test          TRUEtestFALSE
complete      TRUEvalidObjectFALSE
Class
ClassDef      getClassDef
method        NULLObjectvalidObjectTRUEvalidObject
where
```

```
completeTRUEvalidObject
object
```

```
validObjectTRUEtestFALSE
```

```
objectsetValidityobject
validObject
validObject
```

```
setClassclassRepresentation
```

```
setClass("track",
         slots = c(x="numeric", y = "numeric"))
t1 <- new("track", x=1:10, y=sort(stats::rnorm(10)))
## A valid "track" object has the same number of x, y values
validTrackObject <- function(object) {
  if(length(object@x) == length(object@y)) TRUE
  else paste("Unequal x,y lengths: ", length(object@x), ", ",
             length(object@y), sep="")
}
## assign the function as the validity method for the class
setValidity("track", validTrackObject)
## t1 should be a valid "track" object
```



```

validObject(t1)
## Now we do something bad
t2 <- t1
t2@x <- 1:20
## This should generate an error
## Not run: try(validObject(t2))

setClass("trackCurve", contains = "track",
         slots = c(smooth = "numeric"))

## all superclass validity methods are used when validObject
## is called from initialize() with arguments, so this fails
## Not run: trynew("trackCurve", t2)

setClass("twoTrack", slots = c(tr1 = "track", tr2 = "track"))

## validity tests are not applied recursively by default,
## so this object is created (invalidly)
tT <- new("twoTrack", tr2 = t2)

## A stricter test detects the problem
## Not run: try(validObject(tT, complete = TRUE))

```

parallel

parallel-package

"L'Ecuyer-CMRG" [nextRNGStream](#)

[makeForkCluster](#)

library(help = "parallel")

<R-core@r-project.org>

[psnicepskill](#)

clusterApply

```

clusterCall(cl = NULL, fun, ...)
clusterApply(cl = NULL, x, fun, ...)
clusterApplyLB(cl = NULL, x, fun, ...)
clusterEvalQ(cl = NULL, expr)
clusterExport(cl = NULL, varlist, envir = .GlobalEnv)
clusterMap(cl = NULL, fun, ..., MoreArgs = NULL, RECYCLE = TRUE,
           SIMPLIFY = FALSE, USE.NAMES = TRUE,
           .scheduling = c("static", "dynamic"))
clusterSplit(cl = NULL, seq)

```

```

parLapply(cl = NULL, X, fun, ..., chunk.size = NULL)
parSapply(cl = NULL, X, FUN, ..., simplify = TRUE,
          USE.NAMES = TRUE, chunk.size = NULL)
parApply(cl = NULL, X, MARGIN, FUN, ..., chunk.size = NULL)
parRapply(cl = NULL, x, FUN, ..., chunk.size = NULL)
parCapply(cl = NULL, x, FUN, ..., chunk.size = NULL)

```

```

parLapplyLB(cl = NULL, X, fun, ..., chunk.size = NULL)
parSapplyLB(cl = NULL, X, FUN, ..., simplify = TRUE,
            USE.NAMES = TRUE, chunk.size = NULL)

```

cl	NULL
funFUN	
expr	
seq	
varlist	
envir	
x	clusterApplyclusterApplyLBparRapplyparCapply
...	funFUN
MoreArgs	fun
RECYCLE	
X	parLapplyparSapplyparApply
chunk.size	funFUN
MARGIN	
simplifyUSE.NAMES	
	sapply
SIMPLIFY	mapply
.scheduling	

```

clusterCallfun...
clusterEvalQevalqclusterCall
clusterApplyfunx[[1]]...x[[2]]...
clusterApplyLBclusterApplylxpnppclusterApplyLBclusterApply

```

```

clusterMapclusterApplymapplyMapRECYCLEmapplyRECYCLE = TRUESIMPLIFY = TRUEMap
RECYCLE = TRUE
clusterExportvarlist
clusterSplitseq
parLapplyparSapplyparApplylapplysapplyapplyclusterApplyparLapplyLBparSapplyLB
FUNXclusterApplyLBclusterApplyLB
parRapplyparCapplyapplyxparApply
001FUNfun

```

```

clusterCallclusterEvalQclusterSplit
clusterApplyclusterApplyLBx
clusterMapmapply
clusterExport
parLapplyX
parSapplyparApplysapplyapply
parRapplyparCapplyFUN

```

```

parLapplyXxlmapplyparSapplysapply

```

```

## Use option cl.cores to choose an appropriate cluster size.
cl <- makeCluster(getOption("cl.cores", 2))

```

```

clusterApply(cl, 1:2, get("+"), 3)
xx <- 1
clusterExport(cl, "xx")
clusterCall(cl, function(y) xx + y, 2)

```

```

## Use clusterMap like an mapply example
clusterMap(cl, function(x, y) seq_len(x) + y,
           c(a = 1, b = 2, c = 3), c(A = 10, B = 0, C = -10))

```

```

parSapply(cl, 1:20, get("+"), 3)

```

```

## A bootstrapping example, which can be done in many ways:
clusterEvalQ(cl, {
  ## set up each worker. Could also use clusterExport()
  library(boot)
  cd4.rg <- function(data, mle) MASS::mvrnorm(nrow(data), mle$m, mle$v)
  cd4.mle <- list(m = colMeans(cd4), v = var(cd4))

```

```

      NULL
    })
    res <- clusterEvalQ(cl, boot(cd4, corr, R = 100,
                                sim = "parametric", ran.gen = cd4.rg, mle = cd4.mle))

    library(boot)
    cd4.boot <- do.call(c, res)
    boot.ci(cd4.boot, type = c("norm", "basic", "perc"),
            conf = 0.9, h = atanh, hinv = tanh)
    stopCluster(cl)

    ## or
    library(boot)
    run1 <- function(...) {
      library(boot)
      cd4.rg <- function(data, mle) MASS::mvrnorm(nrow(data), mle$m, mle$v)
      cd4.mle <- list(m = colMeans(cd4), v = var(cd4))
      boot(cd4, corr, R = 500, sim = "parametric",
            ran.gen = cd4.rg, mle = cd4.mle)
    }
    cl <- makeCluster(mc <- getOption("cl.cores", 2))
    ## to make this reproducible
    clusterSetRNGStream(cl, 123)
    cd4.boot <- do.call(c, parLapply(cl, seq_len(mc), run1))
    boot.ci(cd4.boot, type = c("norm", "basic", "perc"),
            conf = 0.9, h = atanh, hinv = tanh)
    stopCluster(cl)

```

detectCores

```
detectCores(all.tests = FALSE, logical = TRUE)
```

```

all.tests
logical      FALSE TRUE

```

```
detectCores(TRUE)
```

```
NA
```

```

logical = FALSE logical = TRUE detectCores(logical = FALSE) detectCores(logical =
TRUE)
logical = TRUE

```

`mc.coresmclapplymakeClusterNA`

`detectCores()`
`detectCores(logical = FALSE)`

`makeCluster`

`makeCluster(spec, type, ...)`
`makePSOCKcluster(names, ...)`
`makeForkCluster(nnodes = getOption("mc.cores", 2L), ...)`

`stopCluster(cl = NULL)`

`setDefaultCluster(cl = NULL)`
`getDefaultCluster()`

`registerClusterType(type, starter, make.default = FALSE)`

`spec`
`names` `localhost`
`nnodes`
`type` `registerClusterType`
`...`
`cl` `"cluster"`
`starter` `type`
`make.default` `TRUEmakeClusterFALSE`

`makeCluster` `"PSOCK"` `makePSOCKcluster` `"FORK"` `makeForkCluster`
`makePSOCKcluster` `makeSOCKclusterRscript`
`makeForkCluster`

`makePSOCKcluster`
`master` `somename.local`
`port` `R_PARALLEL_PORT11000:11999`

```

timeout
setup_timeout
outfile stdoutstderr"/dev/nullnul:
homogeneous
rscript
rscript_args Rscript--no-environ
renice 15psnice
rshcmd ssh
user
manual
methods
useXDR
setup_strategy "parallel""PSOCK"manual = FALSEsetup_strategy = "sequential"

makeForkClusterporttimeoutoutfileuseXDR = FALSE"FORK"mcfork
stopCluster
setDefaultClusterssetDefaultCluster(NULL)
registerClusterTypemakeClustertypestarter

c("SOCKcluster", "cluster")
NULL
registerClusterTypetype

homogeneous = TRUE
homogeneous = TRUERscriptrscriptRscript
homogeneous = FALSERscript

```

mcaffinity

mcaffinity

mcaffinity(affinity = NULL)

affinity NULL

mcaffinityaffinity = NULLmcaffinity()NULL

NULLaffinity

[mcparallel](#)

mcchildren

children(select)
readChild(child)
readChildren(timeout = 0)
selectChildren(children = NULL, timeout = 0)
sendChildStdin(child, what)
sendMaster(what, raw.asis = TRUE)

mckill(process, signal = 2L)

select	select
child	"childProcess"
timeout	
children	NULLNULL
what	sendChildStdin
	sendMaster
	what
raw.asis	TRUEwhat
process	process
signal	tools::SIGTERM

```

children
readChildsendMaster
selectChildren
readChildren
sendChildStdinchild
sendMaster
mckillpskill

```

```

children"process"
readChildreadChildren"pid"NULLreadChildren
selectChildrenTRUEFALSENULL
sendChildStdinTRUEchild
sendMasterTRUE
mckillTRUE

```

sendMasterreadChildsendChildStdin $2^{31} - 1$

mcforkmcparallel

```

## Not run:
p <- mcparallel(scan(n = 1, quiet = TRUE))
sendChildStdin(p, "17.4\n")
mccollect(p)[[1]]

## End(Not run)

```

mcfork

mcfork

mcexit

mcfork(estranged = FALSE)

mcexit(exit.code = 0L, send = NULL)

estranged TRUE

exit.code 0L1L

send NULL[sendMaster](#)

mcforkfork[sendMaster](#)stdin[sendChildStdin](#)

fork

forkmcfork

mcexit[sendNULLSIGUSR1parallel::rmChild](#)

mcfork"childProcess""masterProcess""process"estrangedTRUE"estrangedProcess"pid
fd

mcexit

mcforkmcparallelmclapplypvecR.app

Tcl

mcforkmcforkmcforkforkexec

[makeClusterclusterApply](#)

[mcparallelsendMaster](#)

```
## This will work when run as an example, but not when pasted in.
p <- parallel::mcfork()
if (inherits(p, "masterProcess")) {
  cat("I'm a child! ", Sys.getpid(), "\n")
  parallel::mcexit("I was a child")
}
cat("I'm the master\n")
unserialize(parallel::readChildren(1.5))
```

mclapply

lapplymapply

mclapplylapplyXFUNX

mc.cores = 1

mcmapplymapplymcMapMap

```
mclapply(X, FUN, ...,
  mc.preschedule = TRUE, mc.set.seed = TRUE,
  mc.silent = FALSE, mc.cores = getOption("mc.cores", 2L),
  mc.cleanup = TRUE, mc.allow.recursive = TRUE, affinity.list = NULL)
```

```
mcmapply(FUN, ...,
  MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE,
  mc.preschedule = TRUE, mc.set.seed = TRUE,
  mc.silent = FALSE, mc.cores = getOption("mc.cores", 2L),
  mc.cleanup = TRUE, affinity.list = NULL)
```

mcMap(f, ...)

X as.list

FUN mclapplyXmcmapply...

f ...

... mclapplyFUNmcmapplymcMapmapply

MoreArgsSIMPLIFYUSE.NAMES

mapply

mc.preschedule TRUEFALSEXXXmc.cores

mc.set.seed mcparallel

mc.silent TRUEstdoutstderr

mc.cores MC_CORES

mc.cleanup TRUESIGTERMmclapplymclapplyFALSESIGTERM

mc.allow.recursive

mclapply

affinity.list Xmcaffinitymc.preschedule = FALSE

```

mclapplylapplymc.cores > 1mc.cores == 1affinity.listNULLlapply
mc.preschedule = TRUEX
Xmc.cores
lapplymcparallelmc.preschedule = TRUE
ulimit -n"unable to create a pipe"
 $2^{31} - 1$ 
affinity.listXXaffinity.listmc.core

```

```

mclapplyXX
mcmapplymapply
mcMap
try(..., silent = TRUE)"try-error"X"try-error"NULLFUNNULLmcmappl"try-error"
SIMPLIFYFALSE

```

```

R.app
Tcl
mcforkmakeClusterclusterApply

```

```

affinity.list

```

```

mcparallelpvecparLapplyclusterMap
simplify2arraysapply

```

```

simplify2array(mclapply(rep(4, 5), rnorm))
# use the same random numbers for all values
set.seed(1)
simplify2array(mclapply(rep(4, 5), rnorm, mc.preschedule = FALSE,
                        mc.set.seed = FALSE))

## Contrast this with the examples for clusterCall
library(boot)
cd4.rg <- function(data, mle) MASS::mvrnorm(nrow(data), mle$m, mle$v)
cd4.mle <- list(m = colMeans(cd4), v = var(cd4))
mc <- getOption("mc.cores", 2)
run1 <- function(...) boot(cd4, corr, R = 500, sim = "parametric",
                          ran.gen = cd4.rg, mle = cd4.mle)
## To make this reproducible:
set.seed(123, "L'Ecuyer")
res <- mclapply(seq_len(mc), run1)
cd4.boot <- do.call(c, res)
boot.ci(cd4.boot, type = c("norm", "basic", "perc"),

```

```

        conf = 0.9, h = atanh, hinv = tanh)

## Usage of the affinity.list parameter
A <- runif(2500000,0,100)
B <- runif(2500000,0,100)
C <- runif(5000000,0,100)
first <- function(i) head(sort(i), n = 1)

# Restrict all elements of X to run on CPU 1 and 2
affL <- list(c(1,2), c(1,2), c(1,2))
mclapply(list(A, A, A), first, mc.preschedule = FALSE, affinity.list = affL)

# Completion times are assumed to have a high variance
# To optimize the overall execution time elements of X are scheduled to suitable CPUs
# Assuming that the runtime for C is as long as the runtime of A plus B
# mapping: A to 1 , B to 1, C to 2
X <- list(A, B, C)
affL <- c(1, 1, 2)
mclapply(X, first, mc.preschedule = FALSE, affinity.list = affL)

```

mcparallel

```

mcparallel
mccollect

```

```

mcparallel(expr, name, mc.set.seed = TRUE, silent = FALSE,
            mc.affinity = NULL, mc.interactive = FALSE,
            detached = FALSE)

mccollect(jobs, wait = TRUE, timeout = 0, intermediate = FALSE)

```

expr	mcfork mcparallel NULL mccollect
name	
mc.set.seed	
silent	TRUE
mc.affinity	NULL
mc.interactive	TRUEFALSENA
detached	TRUE
jobs	jobscollect
wait	FALSEtimeout
timeout	waitFALSE
intermediate	FALSEcollect

```

mcparallelexprmccollect
mccollectwaitTRUEcollectwaitFALSEmccollectjobs
exprsendMastermccollectNULL
mcparallelmccollectmccollectwait = TRUEmccollectwait = FALSEmccollect
mc.affinity

```

```

mcparallel"parallelJob""childProcess"mcforknamename
mccollectNULL

```

```

mc.set.seed = FALSE.Random.seed
mc.set.seed      =      TRUE RNGkind("L'Ecuyer-CMRG")nextRNGStreammc.reset.stream
mcparallel

```

$2^{31} - 1$

[pvecmclapply](#)

```

p <- mcparallel(1:10)
q <- mcparallel(1:20)
# wait for both jobs to finish and collect all results
res <- mccollect(list(p, q))

```

```

p <- mcparallel(1:10)
mccollect(p, wait = FALSE, 10) # will retrieve the result (since it's fast)
mccollect(p, wait = FALSE)     # will signal the job as terminating
mccollect(p, wait = FALSE)     # there is no longer such a job

```

```

# a naive parallel lapply can be created using mcparallel alone:
jobs <- lapply(1:10, function(x) mcparallel(rnorm(x), name = x))
mccollect(jobs)

```

pvec

pvec

mc.cores = 1

pvec(v, FUN, ..., mc.set.seed = TRUE, mc.silent = FALSE,
 mc.cores = getOption("mc.cores", 2L), mc.cleanup = TRUE)

v

FUN

... FUN

mc.set.seed [mcparallel](#)

mc.silent TRUEstdoustderr

mc.cores MC_CORES

mc.cleanup [mclapply](#)

pvecFUN(x, ...)FUNxFUNFUN

pvec[mclapply](#)mclapplyFUNpvecc(FUN(x[1]), FUN(x[2]))FUN(x[1:2])FUNmclapplyFUN

mc.cores == 1FUN(v, ...)

v

pvec

[mcfork](#)

[mcparallel](#)[mclapply](#)[parLapply](#)[clusterMap](#)

```

x <- pvec(1:1000, sqrt)
stopifnot(all(x == sqrt(1:1000)))

# One use is to convert date strings to unix time in large datasets
# as that is a relatively slow operation.
# So let's get some random dates first
# (A small test only with 2 cores: set options("mc.cores")
# and increase N for a larger-scale test.)
N <- 1e5
dates <- sprintf('%04d-%02d-%02d', as.integer(2000+rnorm(N)),
                 as.integer(runif(N, 1, 12)), as.integer(runif(N, 1, 28)))

system.time(a <- as.POSIXct(dates))

# But specifying the format is faster
system.time(a <- as.POSIXct(dates, format = "%Y-%m-%d"))

# pvec ought to be faster, but system overhead can be high
system.time(b <- pvec(dates, as.POSIXct, format = "%Y-%m-%d"))
stopifnot(all(a == b))

# using mclapply for this would much slower because each value
# will require a separate call to as.POSIXct()
# as lapply(dates, as.POSIXct) does
system.time(c <- unlist(mclapply(dates, as.POSIXct, format = "%Y-%m-%d")))
stopifnot(all(a == c))

```

RNGstreams

```

nextRNGStream(seed)
nextRNGSubStream(seed)

clusterSetRNGStream(cl = NULL, iseed)
mc.reset.stream()

seed          .Random.seed"L'Ecuyer-CMRG"RNG
cl            NULL
iseed         set.seedNULL

RNGkind("L'Ecuyer-CMRG")21912127276
.Random.seed
clusterSetRNGStream"L'Ecuyer-CMRG"set.seed(iseed)
mc.reset.stream()mcparallel(mc.set.seed = TRUE)mclapplypvec

```



```
nextRNGStreamnextRNGSubStream.Random.seed
```

RNG

```
RNGkind("L'Ecuyer-CMRG")
set.seed(123)
(s <- .Random.seed)
## do some work involving random numbers.
nextRNGStream(s)
nextRNGSubStream(s)
```

splitIndices

1:nxncl

splitIndices(nx, ncl)

nx
ncl

ncl

splitIndices(20, 3)

splines

splines-package

[bsns](#)

```
library(help = "splines")
```

```
<bates@stat.wisc.edu><Bill.Venables@csiro.au>  
<R-core@r-project.org>
```

asVector

asVector(object)

object

asVectoras.vectoras.vectorasVectorxyVector

xyVector

```
require(stats)
ispl <- interpSpline( weight ~ height, women )
pred <- predict(ispl)
class(pred)
utils::str(pred)
asVector(pred)
```

backSpline

backSpline(object)

object nbSplinepolySpline

polySplineobject

interpSpline

```
require(graphics)
ispl <- interpSpline( women$height, women$weight )
bspl <- backSpline( ispl )
plot( bspl )                    # plots over the range of the knots
points( women$weight, women$height )
```

bs

```
bs(x, df = NULL, knots = NULL, degree = 3, intercept = FALSE,
   Boundary.knots = range(x), warn.outside = TRUE)
```

```
x
df          dfknotsbs()df-degreexNULLlength(knots)df = degree - intercept
knots       NULLBoundary.knots
degree      3
intercept   TRUEFALSE
Boundary.knots NAknotsBoundary.knotsxBoundary.knots
warn.outside logicalwarningx
```

```
bssplineDesignx
Boundary.knotsrange(x)bs() $\leq$ 
formulasubset = *xbs(x)model.frame
```

```
c(length(x), df)dfknotsdf = length(knots) + degreebsknotsBoundary.knots
predict.bs()
```

```
Boundary.knots
```

```
nspolysmooth.splinepredict.bsSafePrediction
```

```
require(stats); require(graphics)
bs(women$height, df = 5)
summary(fm1 <- lm(weight ~ bs(height, df = 5), data = women))
```

```
## example of safe prediction
plot(women, xlab = "Height (in)", ylab = "Weight (lb)")
ht <- seq(57, 73, length.out = 200)
lines(ht, predict(fm1, data.frame(height = ht)))
```

interpSpline

xydefaultformuladata.frameformula

```
interpSpline(obj1, obj2, bSpline = FALSE, period = NULL,  
             ord = 4L,  
             na.action = na.fail, sparse = FALSE)
```

obj1	x
obj2	obj1obj1
bSpline	TRUEFALSE
period	
ord	$ord = d + 1$ $d = 4$
na.action	NAna.omitna.failinterpSpline
sparse	splineDesign

splinenbSplinenpolySpline

[splineKnotssplineOrderperiodicSpline](#)

```
require(graphics); require(stats)  
ispl <- interpSpline( women$height, women$weight )  
ispl2 <- interpSpline( weight ~ height, women )  
# ispl and ispl2 should be the same  
plot( predict( ispl, seq( 55, 75, length.out = 51 ) ), type = "l" )  
points( women$height, women$weight )  
plot( ispl )      # plots over the range of the knots  
points( women$height, women$weight )  
splineKnots( ispl )
```

ns

```
ns(x, df = NULL, knots = NULL, intercept = FALSE,
   Boundary.knots = range(x))
```

```
x
df      dfns()df - 1 - interceptxdf = NULLlength(knots)
knots    xBoundary.knots
intercept TRUEFALSE
Boundary.knots knotsBoundary.knotsxBoundary.knots
```

[nssplineDesign](#)

```
subset = *xns(x)model.frame
```

```
length(x) * dfdfknotsdf = length(knots) + 1 + interceptnsknotsBoundary.knots
predict.ns()
```

[bspredict.nsSafePrediction](#)

```
require(stats); require(graphics)
ns(women$height, df = 5)
summary(fm1 <- lm(weight ~ ns(height, df = 5), data = women))

## To see what knots were selected
attr(terms(fm1), "predvars")

## example of safe prediction
plot(women, xlab = "Height (in)", ylab = "Weight (lb)")
ht <- seq(57, 73, length.out = 200) ; nD <- data.frame(height = ht)
lines(ht, p1 <- predict(fm1, nD))
stopifnot(all.equal(p1, predict(update(fm1, . ~
                                   splines::ns(height, df=5)), nD)))
# not true in R < 3.5.0
```

periodicSpline

xy

```
periodicSpline(obj1, obj2, knots, period = 2*pi, ord = 4L)
```

obj1	x
obj2	obj1obj1
knots	
period	2 * pi
ord	splineOrder

[splinepbSplineppolySpline](#)

[splineKnotsinterpSpline](#)

```
require(graphics); require(stats)
xx <- seq( -pi, pi, length.out = 16 )[-1]
yy <- sin( xx )
frm <- data.frame( xx, yy )
pispl <- periodicSpline( xx, yy, period = 2 * pi )
pispl
pispl2 <- periodicSpline( yy ~ xx, frm, period = 2 * pi )
stopifnot(all.equal(pispl, pispl2)) # pispl and pispl2 are the same

plot( pispl )          # displays over one period
points( yy ~ xx, col = "brown" )
plot( predict( pispl, seq(-3*pi, 3*pi, length.out = 101) ), type = "l" )
```

polySpline

```
polySpline(object, ...)  
as.polySpline(object, ...)
```

```
object      spline  
...
```

polySpline

[interpSpline](#)[periodicSplines](#)[splineKnots](#)[splineOrder](#)

```
require(graphics)  
ispl <- polySpline(interpSpline( weight ~ height,  women, bSpline = TRUE))  
  
print( ispl )    # print the piecewise polynomial representation  
  
plot( ispl )     # plots over the range of the knots  
points( women$height, women$weight )
```

predict.bs

```
## S3 method for class 'bs'  
predict(object, newx, ...)
```

```
## S3 method for class 'ns'  
predict(object, newx, ...)
```

```
object      bsnsknotsdegree  
newx        x  
...
```



```
objectx
predict"bs""ns"predict
```

```
bsnspoly
```

```
require(stats)
basis <- ns(women$height, df = 5)
newX <- seq(58, 72, length.out = 51)
# evaluate the basis at the new data
predict(basis, newX)
```

```
predict.bSpline
```

```
predictbSplinepolySplineplotpredictxyVectortype = "l"
```

```
## S3 method for class 'bSpline'
predict(object, x, nseg = 50, deriv = 0, ...)
## S3 method for class 'nbSpline'
predict(object, x, nseg = 50, deriv = 0, ...)
## S3 method for class 'pbSpline'
predict(object, x, nseg = 50, deriv = 0, ...)
## S3 method for class 'npolySpline'
predict(object, x, nseg = 50, deriv = 0, ...)
## S3 method for class 'ppolySpline'
predict(object, x, nseg = 50, deriv = 0, ...)
```

```
object      bSplinepolySpline
x           xxnseq
nseg        xobjectx
deriv       splineOrder(object) - 1
...
```

```
xyVector
```

```
x           x
y           derivx
```

xyVectorinterpSplineperiodicSpline

```
require(graphics); require(stats)
ispl <- interpSpline( weight ~ height, women )
opar <- par(mfrow = c(2, 2), las = 1)
plot(predict(ispl, nseg = 201),      # plots over the range of the knots
      main = "Original data with interpolating spline", type = "l",
      xlab = "height", ylab = "weight")
points(women$height, women$weight, col = 4)
plot(predict(ispl, nseg = 201, deriv = 1),
      main = "First derivative of interpolating spline", type = "l",
      xlab = "height", ylab = "weight")
plot(predict(ispl, nseg = 201, deriv = 2),
      main = "Second derivative of interpolating spline", type = "l",
      xlab = "height", ylab = "weight")
plot(predict(ispl, nseg = 401, deriv = 3),
      main = "Third derivative of interpolating spline", type = "l",
      xlab = "height", ylab = "weight")
par(opar)
```

splineDesign

knotsx

```
splineDesign(knots, x, ord = 4, derivs, outer.ok = FALSE,
             sparse = FALSE)
spline.des   (knots, x, ord = 4, derivs, outer.ok = FALSE,
             sparse = FALSE)
```

knots

x outer.okxknots[ord]knots[length(knots) - (ord-1)]

ord

derivs 0ord - 1xxx

outer.ok xx

sparse "sparseMatrix"

length(x)length(knots) - ordknotxordlength(knots) - ord

spline.desknotsordderivsdesignndesignsplineDesign

```

require(graphics)
splineDesign(knots = 1:10, x = 4:7)
splineDesign(knots = 1:10, x = 4:7, derivs = 1)
## visualize band structure
Matrix::drop0(zapsmall(6*splineDesign(knots = 1:40, x = 4:37, sparse = TRUE)))

knots <- c(1,1.8,3.5,6.5,7,8.1,9.2,10) # 10 => 10-4 = 6 Basis splines
x <- seq(min(knots)-1, max(knots)+1, length.out = 501)
bb <- splineDesign(knots, x = x, outer.ok = TRUE)

plot(range(x), c(0,1), type = "n", xlab = "x", ylab = "",
      main = "B-splines - sum to 1 inside inner knots")
mtext(expression(B[j](x) * " and " * sum(B[j](x), j == 1, 6)), adj = 0)
abline(v = knots, lty = 3, col = "light gray")
abline(v = knots[c(4,length(knots)-3)], lty = 3, col = "gray10")
lines(x, rowSums(bb), col = "gray", lwd = 2)
matlines(x, bb, ylim = c(0,1), lty = 1)

```

splineKnots

splineKnots(object)

object "spline"

```

ispl <- interpSpline( weight ~ height, women )
splineKnots( ispl )

```

splineOrder

splineOrder(object)

object "spline"

[splineKnots](#)[interpSpline](#)[periodicSpline](#)

splineOrder(interpSpline(weight ~ height, women))

xyVector

xyVector

y
predict.splinuxyVector

xyVector(x, y)

x
y x

xyVector

x
y x

```
require(stats); require(graphics)
ispl <- interpSpline( weight ~ height, women )
weights <- predict( ispl, seq( 55, 75, length.out = 51 ))
class( weights )
plot( weights, type = "l", xlab = "height", ylab = "weight" )
points( women$height, women$weight )
weights
```

stats

stats-package

```
library(help = "stats")
```

```
<R-core@r-project.org>
```

.checkMFClasses

```
.checkMFClasses
.MFclass
.getXlevels()factorcharacter

.checkMFClasses(cl, m, ordNotOK = FALSE)
.MFclass(x)
.getXlevels(Terms, m)

cl
m          model.frame\(\)
x
ordNotOK
Terms      termsterms.object
```

```
model.matrix()rpart
```

```
.checkMFClasses()stop()NULL  
.MFclass()"logical""ordered""factor""numeric""nmatrix.*""other"  
.getXlevelslistNULL
```

```
sapply(warpbreaks, .MFclass) # "numeric" plus 2 x "factor"  
sapply(iris, .MFclass) # 4 x "numeric" plus "factor"  
  
mf <- model.frame(Sepal.Width ~ Species, iris)  
mc <- model.frame(Sepal.Width ~ Sepal.Length, iris)  
  
.checkMFClasses("numeric", mc) # nothing else  
.checkMFClasses(c("numeric", "factor"), mf)  
  
## simple .getXlevels() cases :  
(xl <- .getXlevels(terms(mf), mf)) # a list with one entry " $ Species" with 3 levels:  
stopifnot(exprs = {  
  identical(xl$Species, levels(iris$Species))  
  identical(.getXlevels(terms(mc), mc), xl[0]) # a empty named list, as no factors  
  is.null(.getXlevels(terms(x~x), list(x=1)))  
})
```

acf

acfpacfccf

```
acf(x, lag.max = NULL,  
    type = c("correlation", "covariance", "partial"),  
    plot = TRUE, na.action = na.fail, demean = TRUE, ...)  
  
pacf(x, lag.max, plot, na.action, ...)  
  
## Default S3 method:  
pacf(x, lag.max = NULL, plot = TRUE, na.action = na.fail,  
     ...)  
  
ccf(x, y, lag.max = NULL, type = c("correlation", "covariance"),  
    plot = TRUE, na.action = na.fail, ...)  
  
## S3 method for class 'acf'  
x[i, j]
```

```

xy          ccf"acf"
lag.max     10log10(N/m)Nm
type        "correlation""covariance""partial"
plot        TRUE
na.action   na.pass
demean
...         plot.acf
i
j

```

```

type"correlation""covariance"
na.actionna.pass
lag.max
plot"acf"

```

```

print"acf"

```

```

"acf"

```

```

lag
acf      lag
type     type
n.used
series   x
snames

```

```

kccf(x, y)x[t+k]y[t]
plotTRUE

```

```

pacf

```

```

plot.acfARMAacf

```



```
require(graphics)

## Examples from Venables & Ripley
acf(lh)
acf(lh, type = "covariance")
pacf(lh)

acf(ldeaths)
acf(ldeaths, ci.type = "ma")
acf(ts.union(mdeaths, fdeaths))
ccf(mdeaths, fdeaths, ylab = "cross-correlation")
# (just the cross-correlations)

presidents # contains missing values
acf(presidents, na.action = na.pass)
pacf(presidents, na.action = na.pass)
```

acf2AR

acf2AR(acf)

acf

1 <= p <= length(acf)

[ARMAacf.ar.yw](#)

```
(Acf <- ARMAacf(c(0.6, 0.3, -0.2)))
acf2AR(Acf)
```

add1

scope

```

add1(object, scope, ...)

## Default S3 method:
add1(object, scope, scale = 0, test = c("none", "Chisq"),
      k = 2, trace = FALSE, ...)

## S3 method for class 'lm'
add1(object, scope, scale = 0, test = c("none", "Chisq", "F"),
      x = NULL, k = 2, ...)

## S3 method for class 'glm'
add1(object, scope, scale = 0,
      test = c("none", "Rao", "LRT", "Chisq", "F"),
      x = NULL, k = 2, ...)

drop1(object, scope, ...)

## Default S3 method:
drop1(object, scope, scale = 0, test = c("none", "Chisq"),
       k = 2, trace = FALSE, ...)

## S3 method for class 'lm'
drop1(object, scope, scale = 0, all.cols = TRUE,
       test = c("none", "Chisq", "F"), k = 2, ...)

## S3 method for class 'glm'
drop1(object, scope, scale = 0,
       test = c("none", "Rao", "LRT", "Chisq", "F"),
       k = 2, ...)

```

```

object
scope
scale       $C_p$  0 NULL
test       lmaovglm  $\chi^2$  lmgglm "LRT" "Rao" "Chisq"
k           $C_p$ 
trace      TRUE
x          add1
all.cols   FALSE
...

```

```

drop1scope
scope.
lmgglmfit
 $2ppC_pRSS/scale + 2p - nC_pC_pAIC$ 
"glm" anova.glm

```

"anova"

na.action = na.omit

nobs

keep

C_p

stepaovlmextractAICanova

```
require(graphics); require(utils)
## following example(swiss)
lm1 <- lm(Fertility ~ ., data = swiss)
add1(lm1, ~ I(Education^2) + .^2)
drop1(lm1, test = "F") # So called 'type II' anova

## following example(glm)

drop1(glm.D93, test = "Chisq")
drop1(glm.D93, test = "F")
add1(glm.D93, scope = ~outcome*treatment, test = "Rao") ## Pearson Chi-square
```

addmargins

addmargins(A, margin = seq_along(dim(A)), FUN = sum, quiet = FALSE)

A	"dim""dimnames"A
margin	margin
FUN	listmarginfunction
quiet	

FUN

`tablearray` `AmarginFUN`

<https://BendixCarstensen.com>

`table` `tablemargin.table`

```
Aye <- sample(c("Yes", "Si", "Oui"), 177, replace = TRUE)
Bee <- sample(c("Hum", "Buzz"), 177, replace = TRUE)
Sea <- sample(c("White", "Black", "Red", "Dead"), 177, replace = TRUE)
(A <- table(Aye, Bee, Sea))
(aA <- addmargins(A))

ftable(A)
ftable(aA)

# Non-commutative functions - note differences between resulting tables:
ftable( addmargins(A, c(3, 1),
  FUN = list(list(Min = min, Max = max),
    Sum = sum)))
ftable( addmargins(A, c(1, 3),
  FUN = list(Sum = sum,
    list(Min = min, Max = max))))

# Weird function needed to return the N when computing percentages
sqsm <- function(x) sum(x)^2/100
B <- table(Sea, Bee)
round(sweep(addmargins(B, 1, list(list(All = sum, N = sqsm))), 2,
  apply(B, 2, sum)/100, `/^`), 1)
round(sweep(addmargins(B, 2, list(list(All = sum, N = sqsm))), 1,
  apply(B, 1, sum)/100, `/^`), 1)

# A total over Bee requires formation of the Bee-margin first:
mB <- addmargins(B, 2, FUN = list(list(Total = sum)))
round(ftable(sweep(addmargins(mB, 1, list(list(All = sum, N = sqsm))), 2,
  apply(mB, 2, sum)/100, `/^`), 1)

## Zero.Printing table+margins:
set.seed(1)
x <- sample( 1:7, 20, replace = TRUE)
y <- sample( 1:7, 20, replace = TRUE)
tx <- addmargins( table(x, y) )
print(tx, zero.print = ".")
```

aggregate

```
aggregate(x, ...)

## Default S3 method:
aggregate(x, ...)

## S3 method for class 'data.frame'
aggregate(x, by, FUN, ..., simplify = TRUE, drop = TRUE)

## S3 method for class 'formula'
aggregate(x, data, FUN, ...,
          subset, na.action = na.omit)

## S3 method for class 'ts'
aggregate(x, nfrequency = 1, FUN = sum, ndeltat = 1,
          ts.eps = getOption("ts.eps"), ...)
```

x	formula	<code>formula</code>	<code>y ~ x</code>	<code>cbind(y1, y2) ~ x1 + x2</code>	<code>y</code>
by	x				
FUN					
simplify					
drop	drop=FALSE				
data					
subset					
na.action	NA				
nfrequency	x				
ndeltat	x				
ts.eps	nfrequency				
...					

```
aggregate
aggregate.defaultxx
aggregate.data.framexxbyFUN...byxbyxxsimplifybyFUN
aggregate.data.framebyaggregate(x, by, FUN)aggregate(by, x, FUN)x
aggregate.tsFUNxxfrequency(x) / nfrequencyFUN...nfrequency
FUNmatch.fun
```

```
"ts"c("mts", "ts")
byxbyGroup.by[[]]
```

```
"formula"formulax
```

[applylapplytapply](#)

```
## Compute the averages for the variables in 'state.x77', grouped
## according to the region (Northeast, South, North Central, West) that
## each state belongs to.
aggregate(state.x77, list(Region = state.region), mean)
```

```
## Compute the averages according to region and the occurrence of more
## than 130 days of frost.
aggregate(state.x77,
          list(Region = state.region,
               Cold = state.x77["Frost"] > 130),
          mean)
## (Note that no state in 'South' is THAT cold.)
```

```
## example with character variables and NAs
testDF <- data.frame(v1 = c(1,3,5,7,8,3,5,NA,4,5,7,9),
                     v2 = c(11,33,55,77,88,33,55,NA,44,55,77,99) )
by1 <- c("red", "blue", 1, 2, NA, "big", 1, 2, "red", 1, NA, 12)
by2 <- c("wet", "dry", 99, 95, NA, "damp", 95, 99, "red", 99, NA, NA)
aggregate(x = testDF, by = list(by1, by2), FUN = "mean")
```

```
# and if you want to treat NAs as a group
fby1 <- factor(by1, exclude = "")
fby2 <- factor(by2, exclude = "")
aggregate(x = testDF, by = list(fby1, fby2), FUN = "mean")
```

```
## Formulas, one ~ one, one ~ many, many ~ one, and many ~ many:
aggregate(weight ~ feed, data = chickwts, mean)
aggregate(breaks ~ wool + tension, data = warpbreaks, mean)
aggregate(cbind(Ozone, Temp) ~ Month, data = airquality, mean)
aggregate(cbind(ncases, ncontrols) ~ alcgp + tobgp, data = esoph, sum)
```

```
## "complete cases" vs. "available cases"
colSums(is.na(airquality)) # NAs in Ozone but not Temp
## the default is to summarize *complete cases*:
```

```

aggregate(cbind(Ozone, Temp) ~ Month, data = airquality, FUN = mean)
## to handle missing values *per variable*:
aggregate(cbind(Ozone, Temp) ~ Month, data = airquality, FUN = mean,
          na.action = na.pass, na.rm = TRUE)

## Dot notation:
aggregate(. ~ Species, data = iris, mean)
aggregate(len ~ ., data = ToothGrowth, mean)

## Often followed by xtabs():
ag <- aggregate(len ~ ., data = ToothGrowth, mean)
xtabs(len ~ ., data = ag)

## Formula interface via 'by' (for pipe operations)
ToothGrowth |> aggregate(len ~ ., FUN = mean)

## Compute the average annual approval ratings for American presidents.
aggregate(presidents, nfrequency = 1, FUN = mean)
## Give the summer less weight.
aggregate(presidents, nfrequency = 1,
          FUN = weighted.mean, w = c(1, 1, 0.5, 1))

```

AIC

$$-2 + kn_{par}n_{par}k = 2k = \log(n)n$$

AIC(object, ..., k = 2)

BIC(object, ...)

object	logLiklogLik
...	
k	k = 2

[logLiklogLiklogLik](#)

[extractAICAIC"lm"extractAIC](#)

BICAIC(object, ..., k = log(nobs(object)))"nobs"logLiknobsNA

k

data.framedf

extractAIClogLiknobs

```
lm1 <- lm(Fertility ~ . , data = swiss)
AIC(lm1)
stopifnot(all.equal(AIC(lm1),
                    AIC(logLik(lm1))))
BIC(lm1)

lm2 <- update(lm1, . ~ . -Examination)
AIC(lm1, lm2)
BIC(lm1, lm2)
```

alias

```
alias(object, ...)
```

```
## S3 method for class 'formula'
alias(object, data, ...)
```

```
## S3 method for class 'lm'
alias(object, complete = TRUE, partial = FALSE,
      partial.pattern = FALSE, ...)
```

```
object          lmaovalias.formula
data
complete
partial
partial.pattern
```

```
...
```

```
"lm"aliasaov
"lm"
```



```
class"listof"  
  
Model  
Complete  
Partial          "mtable"print
```

```
op <- options(contrasts = c("contr.helmert", "contr.poly"))  
npk.aov <- aov(yield ~ block + N*P*K, npk)  
alias(npk.aov)  
options(op) # reset
```

anova

```
anova(object, ...)
```

```
object          lmg1m  
...
```

```
anova  
anova  
anova
```

```
na.action = na.omit
```

```
coefficientseffectsfitted.valuesresidualssummarydrop1add1
```

`anova.glm`

```
## S3 method for class 'glm'
anova(object, ..., dispersion = NULL, test = NULL)
```

```
object...      glmglmobjects"glmmlist"
dispersion
test           "Chisq""LRT""Rao""F""Cp"stat.anovaFALSE
```

```
gaussianquasibinomialquasipoissonanova.glm
dispersiontest=FALSECpσ2"LRT""Rao"
"Chisq"
summary.glm
```

```
"anova""data.frame"
```

```
na.action = na.omitanova
```

```
glm
anova
drop1
```

```
## --- Continuing the Example from '?glm':

anova(glm.D93, test = FALSE)
anova(glm.D93, test = "Cp")
anova(glm.D93, test = "Chisq")
glm.D93a <-
  update(glm.D93, ~treatment*outcome) # equivalent to Pearson Chi-square
anova(glm.D93, glm.D93a, test = "Rao")
```

anova.lm

```
## S3 method for class 'lm'
anova(object, ...)

## S3 method for class 'lmlist'
anova(object, ..., scale = 0, test = "F")
```

```
object...      lmlm
test           "F""Chisq""Cp"NULL
scale           $\sigma^2$ 
```

scale $C_p\sigma^2$

"anova""data.frame"

na.action = na.omitanova.lmlist

[lmanova](#)

[drop1](#)

```
## sequential table
fit <- lm(sr ~ ., data = LifeCycleSavings)
anova(fit)

## same effect via separate models
fit0 <- lm(sr ~ 1, data = LifeCycleSavings)
fit1 <- update(fit0, . ~ . + pop15)
fit2 <- update(fit1, . ~ . + pop75)
fit3 <- update(fit2, . ~ . + dpi)
fit4 <- update(fit3, . ~ . + ddpi)
anova(fit0, fit1, fit2, fit3, fit4, test = "F")

anova(fit4, fit2, fit0, test = "F") # unconventional order
```

anova.mlm

```
## S3 method for class 'mlm'
anova(object, ...,
       test = c("Pillai", "Wilks", "Hotelling-Lawley", "Roy",
                "Spherical"),
       Sigma = diag(nrow = p), T = Thin.row(Proj(M) - Proj(X)),
       M = diag(nrow = p), X = ~0,
       idata = data.frame(index = seq_len(p)), tol = 1e-7)
```

```
object      "mlm"
...          "mlm"
test
Sigma        test == "Spherical"Sigma
T            MX
M
X
idata
tol          qr
```

```
anova.mlm
summary.manova
"Spherical"SigmaF
TMXidataM/X
anova.lm
anova

"anova""data.frame"
```

[summary.manova](#)

```

require(graphics)
utils::example(SSD) # Brings in the mlmfit and reacttime objects

mlmfit0 <- update(mlmfit, ~0)

### Traditional tests of intrasubj. contrasts
## Using MANOVA techniques on contrasts:
anova(mlmfit, mlmfit0, X = ~1)

## Assuming sphericity
anova(mlmfit, mlmfit0, X = ~1, test = "Spherical")

### tests using intra-subject 3x2 design
idata <- data.frame(deg = gl(3, 1, 6, labels = c(0, 4, 8)),
                    noise = gl(2, 3, 6, labels = c("A", "P")))

anova(mlmfit, mlmfit0, X = ~ deg + noise,
      idata = idata, test = "Spherical")
anova(mlmfit, mlmfit0, M = ~ deg + noise, X = ~ noise,
      idata = idata, test = "Spherical" )
anova(mlmfit, mlmfit0, M = ~ deg + noise, X = ~ deg,
      idata = idata, test = "Spherical" )

f <- factor(rep(1:2, 5)) # bogus, just for illustration
mlmfit2 <- update(mlmfit, ~f)
anova(mlmfit2, mlmfit, mlmfit0, X = ~1, test = "Spherical")
anova(mlmfit2, X = ~1, test = "Spherical")
# one-model form, equiv. to previous

### There seems to be a strong interaction in these data
plot(colMeans(reacttime))

```

ansari.test

```

ansari.test(x, ...)

## Default S3 method:
ansari.test(x, y,
            alternative = c("two.sided", "less", "greater"),
            exact = NULL, conf.int = FALSE, conf.level = 0.95,
            ...)

## S3 method for class 'formula'
ansari.test(formula, data, subset, na.action, ...)

```

```

x
y
alternative      "two.sided""greater""less"
exact
conf.int
conf.level
formula          lhs ~ rhs|lhsrhs
data             model.frameformulaenvironment(formula)
subset
na.action        NAgetOption("na.action")
...

```

```

xyf((t - m)/s)/sf(t - m)msss ≠ 1s > 1x"greater"s < 1"less"
exact
s

```

```

"htest"
statistic
p.value
null.value      s
alternative
method          "Ansari-Bradley test"
data.name
conf.int        conf.int = TRUE
estimate        conf.int = TRUE

```

$ss^2 ss^2$

```

fligner.testk mood.testvar.testbartlett.test
ansari_test

```

```
## Hollander & Wolfe (1973, p. 86f):
## Serum iron determination using Hyland control sera
ramsay <- c(111, 107, 100, 99, 102, 106, 109, 108, 104, 99,
            101, 96, 97, 102, 107, 113, 116, 113, 110, 98)
jung.parekh <- c(107, 108, 106, 98, 105, 103, 110, 105, 104,
                 100, 96, 108, 103, 104, 114, 114, 113, 108, 106, 99)
ansari.test(ramsay, jung.parekh)

ansari.test(rnorm(10), rnorm(10, 0, 2), conf.int = TRUE)

## try more points - failed in 2.4.1
ansari.test(rnorm(100), rnorm(100, 0, 2), conf.int = TRUE)
```

aov

`lmError(.)`

`aov(formula, data = NULL, projections = FALSE, qr = TRUE,
 contrasts = NULL, ...)`

formula

data

projections

qr

contrasts ErrorError

... lmsubsetna.actionweights

[lm](#)

`lmprintsummary`

Error

`weightsError`[model.tables](#)

`c("aov", "lm")c("maov", "aov", "mlm", "lm")c("aovlist", "listof")printsummary`

`aov`[lme](#)

[replications](#)

aov

[lmsummary.aovreplicationsaliasprojmodel.tablesTukeyHSD](#)

```
## From Venables and Ripley (2002) p.165.

## Set orthogonal contrasts.
op <- options(contrasts = c("contr.helmert", "contr.poly"))
( npk.aov <- aov(yield ~ block + N*P*K, npk) )
summary(npk.aov)
coefficients(npk.aov)

## to show the effects of re-ordering terms contrast the two fits
aov(yield ~ block + N * P + K, npk)
aov(terms(yield ~ block + N * P + K, keep.order = TRUE), npk)

## as a test, not particularly sensible statistically
npk.aovE <- aov(yield ~ N*P*K + Error(block), npk)
npk.aovE
summary(npk.aovE)
options(op) # reset to previous
```

approxfun

```
approx (x, y = NULL, xout, method = "linear", n = 50,
        yleft, yright, rule = 1, f = 0, ties = mean, na.rm = TRUE)
```

```
approxfun(x, y = NULL,          method = "linear",
          yleft, yright, rule = 1, f = 0, ties = mean, na.rm = TRUE)
```

xy	xy.coords
xout	
method	"linear""constant"
n	xoutnmin(x)max(x)
yleft	xmin(x)rule
yright	xmax(x)rule
rule	min(x)max(x)rule1NA2rule = 2:1
f	method = "constant"y0y1y0f == 0y1f == 1 y0*(1-f)+y1*ff == 0f == 1y
ties	x"ordered" list list("ordered", mean)
na.rm	NA na.rm=FALSENAyruleNAx


```

na.rm(x, y)method = "linear"xtiesyx(x,y)xmeanminmax
ties = "ordered"xlength(x)
tieslistties[[2]]ties[[1]]"ordered"xties = list("ordered", mean)ties = mean
y

```

```

approxxynmethodrule
approxfunx

```

```

approxfun

```

```

splinesplinefun

```

```

require(graphics)

x <- 1:10
y <- rnorm(10)
par(mfrow = c(2,1))
plot(x, y, main = "approx(.) and approxfun(.)")
points(approx(x, y), col = 2, pch = "*")
points(approx(x, y, method = "constant"), col = 4, pch = "*")

f <- approxfun(x, y)
curve(f(x), 0, 11, col = "green2")
points(x, y)
is.function(fc <- approxfun(x, y, method = "const")) # TRUE
curve(fc(x), 0, 10, col = "darkblue", add = TRUE)
## different extrapolation on left and right side :
plot(approxfun(x, y, rule = 2:1), 0, 11,
      col = "tomato", add = TRUE, lty = 3, lwd = 2)

### Treatment of 'NA's -- are kept if na.rm=FALSE :

xn <- 1:4
yn <- c(1,NA,3:4)
xout <- (1:9)/2
## Default behavior (na.rm = TRUE): NA's omitted; extrapolation gives NA
data.frame(approx(xn,yn, xout))
data.frame(approx(xn,yn, xout, rule = 2))# -> *constant* extrapolation
## New (2019-2020) na.rm = FALSE: NA's are "kept"
data.frame(approx(xn,yn, xout, na.rm=FALSE, rule = 2))
data.frame(approx(xn,yn, xout, na.rm=FALSE, rule = 2, method="constant"))

## NA's in x[] are not allowed:
stopifnot(inherits( try( approx(yn,yn, na.rm=FALSE) ), "try-error"))

```

```

## Give a nice overview of all possibilities rule * method * na.rm :
## -----
## extrapolation 'rule's "N" := NA; "C" := Constant :
allapprox <- function(x, y, xout = NULL, ...) {
  if(is.null(xout)) { rx <- range(x, na.rm=TRUE); xout <- seq(rx[1], rx[2], length.out = 25) }
  rules <- list(N=1, C=2, NC=1:2, CN=2:1)
  methods <- c("constant", "linear")
  ry <- sapply(rules, function(R) {
    sapply(methods, function(M)
      sapply(setNames(, c(TRUE,FALSE)), function(na.)
        approx(x, y, xout=xout, method=M, rule=R, na.rm=na., ...) $y),
        simplify="array")
    }, simplify="array")
  names(dimnames(ry)) <- c("x = ", "na.rm", "method", "rule")
  dimnames(ry)[[1]] <- format(xout)
  ftable(aperm(ry, 4:1)) # --> (4 * 2 * 2) x length(xout) = 16 x 9 matrix
}

(ry <- allapprox(xn, yn, xout)) # nice ftable

## Show treatment of 'ties' :

x <- c(2,2:4,4,4,5,5,7,7,7)
y <- c( 1:6, 5:4, 3:1)
(amy <- approx(x, y, xout = x)$y) # warning, can be avoided by specifying 'ties=':
op <- options(warn=2) # warnings would be error
stopifnot(identical(amy, approx(x, y, xout = x, ties=mean)$y))
(ay <- approx(x, y, xout = x, ties = "ordered")$y)
stopifnot(amy == c(1.5,1.5, 3, 5,5,5, 4.5,4.5, 2,2,2),
          ay == c(2, 2, 3, 6,6,6, 4, 4, 1,1,1))
approx(x, y, xout = x, ties = min)$y
approx(x, y, xout = x, ties = max)$y

## 'ties' + NAs -- notably NAs for tied x[], situation as PR#17604

x <- c(2:3, 3:5, 5:7)
y <- c(1,NA,2:4,NA,1,0)
## allapprox() [defined above] for all variants :
(ryN <- allapprox(x, y, xout = seq(2, 7.5, by = 1/2), ties = mean))
str(tbN <- as.table(ryN)) # 4 x 2 x 2 x 12 array
stopifnot(is.na( tbN[, , na.rm="FALSE", format(c(3, 3.5, 5, 5.5))] ))
## 3 and 3.5 gave values in {2, 2.5} instead of NA, in R <= 4.5.2

```

ar

```

ar(x, aic = TRUE, order.max = NULL,
  method = c("yule-walker", "burg", "ols", "mle", "yw"),
  na.action, series, ...)

```

```

ar.burg(x, ...)
## Default S3 method:
ar.burg(x, aic = TRUE, order.max = NULL,
        na.action = na.fail, demean = TRUE, series,
        var.method = 1, ...)
## S3 method for class 'mts'
ar.burg(x, aic = TRUE, order.max = NULL,
        na.action = na.fail, demean = TRUE, series,
        var.method = 1, ...)

ar.yw(x, ...)
## Default S3 method:
ar.yw(x, aic = TRUE, order.max = NULL,
       na.action = na.fail, demean = TRUE, series, ...)
## S3 method for class 'mts'
ar.yw(x, aic = TRUE, order.max = NULL,
       na.action = na.fail, demean = TRUE, series,
       var.method = 1, ...)

ar.mle(x, aic = TRUE, order.max = NULL, na.action = na.fail,
       demean = TRUE, series, ...)

## S3 method for class 'ar'
predict(object, newdata, n.ahead = 1, se.fit = TRUE, ...)

x
aic          logicalTRUEFALSEorder.max
order.max     $N - 110 \log_{10}(N)N$ method = "mle"
method       "yule-walker"
na.action     na.action = na.pass
demean
series        deparse1(substitute(x))
var.method
...
object        ar()
newdata
n.ahead
se.fit

```

$$x_t - \mu = a_1(x_{t-1} - \mu) + \cdots + a_p(x_{t-p} - \mu) + e_t$$

```

arar.ywar.burgar.olsar.mle
aicar.mlear.ywx
ar.burg
arxar.mle

```

```

ar"ar"

order          aic = TRUEorder.max
ar
var.pred
x.mean
x.intercept    ar.olsx - x.mean
aic            -Inf
n.used
n.obs
order.max      order.max
partialacf     order.max
resid          orderorderNAxresid
method         method
series
frequency
call
asy.var.coef   order > 0
predict.arse.fit = TRUEpredse

```

```

ar.mle
method="mle"
xNAarima()method = "ML"

```

```

ar.ywar.mlear.burg

```

```

ar.olsarimaacf2AR
arima.sim

```

```

ar(lh)
ar(lh, method = "burg")
ar(lh, method = "ols")
ar(lh, FALSE, 4) # fit ar(4)

(sunspot.ar <- ar(sunspot.year))
predict(sunspot.ar, n.ahead = 25)
## try the other methods too

ar(ts.union(BJsales, BJsales.lead))
## Burg is quite different here, as is OLS (see ar.ols)
ar(ts.union(BJsales, BJsales.lead), method = "burg")

```

ar.ols

```

ar.ols(x, aic = TRUE, order.max = NULL, na.action = na.fail,
       demean = TRUE, intercept = demean, series, ...)

```

```

x
aic          TRUEFALSEorder.max
order.max    10 log10(N)N
na.action
demean       x
intercept
series       deparse1(substitute(x))
...

```

ar.olsx

$$x_t - \mu = a_0 + a_1(x_{t-1} - \mu) + \cdots + a_p(x_{t-p} - \mu) + e_t$$

a_0 intercept μ demean

aicar.ols

interceptdemean

"ar"

```
order          aic = TRUEorder.max
ar
var.pred
x.mean         demean
x.intercept    x - x.meanintercept
aic            -Inf
n.used
order.max      order.max
partialacf     NULLar
resid          orderorderNaxresid
method         "Unconstrained LS"
series
frequency
call
asy.se.coef
```

ar

```
ar(lh, method = "burg")
ar.ols(lh)
ar.ols(lh, FALSE, 4) # fit ar(4)

ar.ols(ts.union(BJsales, BJsales.lead))

x <- diff(log(EuStockMarkets))
ar.ols(x, order.max = 6, demean = FALSE, intercept = TRUE)
```

arima

```
arima(x, order = c(0L, 0L, 0L),
      seasonal = list(order = c(0L, 0L, 0L), period = NA),
      xreg = NULL, include.mean = TRUE,
      transform.pars = TRUE,
      fixed = NULL, init = NULL,
      method = c("CSS-ML", "ML", "CSS"), n.cond,
      SSinit = c("Gardner1980", "Rossignol2011"),
      optim.method = "BFGS",
      optim.control = list(), kappa = 1e6)
```

x

order (p, d, q)

seasonal frequency(x)listorderperiodorder

xreg x

include.mean TRUE

transform.pars method = "CSS"method = "ML"transform.pars = FALSEfixed

fixed $(\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q, \Phi_1, \dots, \Phi_P, \Theta_1, \dots, \Theta_Q, \mu),$
 $\phi_i, \theta_i, \Phi_i, \Theta_i, \mu, \mu$ include.meanTRUE
fixedNA
transform.parsFALSEtransform.parsTRUEtransform.pars = FALSE

init fixed

method

n.cond

SSinit KalmanLike

optim.method methodoptim

optim.control optim

kappa

$$X_t = a_1 X_{t-1} + \dots + a_p X_{t-p} + e_t + b_1 e_{t-1} + \dots + b_q e_{t-q}$$

include.meanX - mXxreginclude.mean

optimxreg

"Arima"

coef	coef
sigma2	
var.coef	coefvcov
loglik	
arma	
aic	method = "ML"
residuals	
call	
series	x
code	optim
n.cond	
nobs	nobs()BIC
model	KalmanLike

kappa1e4arima0

"ML"

transform.parstrtransform.pars

n.condn.condn.cond

arimaarima0arima0

https://www.r-project.org/doc/Rnews/Rnews_2002-2.pdf

[predict.Arimaarima.simtsdiagarima0ar](#)


```

arima(lh, order = c(1,0,0))
arima(lh, order = c(3,0,0))
arima(lh, order = c(1,0,1))

arima(lh, order = c(3,0,0), method = "CSS")

arima(USAccDeaths, order = c(0,1,1), seasonal = list(order = c(0,1,1)))
arima(USAccDeaths, order = c(0,1,1), seasonal = list(order = c(0,1,1)),
      method = "CSS") # drops first 13 observations.
# for a model with as few years as this, we want full ML

arima(LakeHuron, order = c(2,0,0), xreg = time(LakeHuron) - 1920)

## presidents contains NAs
## graphs in example(acf) suggest order 1 or 3
require(graphics)
(fit1 <- arima(presidents, c(1, 0, 0)))
nobs(fit1)
tsdiag(fit1)
(fit3 <- arima(presidents, c(3, 0, 0))) # smaller AIC
tsdiag(fit3)
BIC(fit1, fit3)
## compare a whole set of models; BIC() would choose the smallest
AIC(fit1, arima(presidents, c(2,0,0)),
    arima(presidents, c(2,0,1)), # <- chosen (barely) by AIC
    fit3, arima(presidents, c(3,0,1)))

## An example of using the 'fixed' argument:
## Note that the period of the seasonal component is taken to be
## frequency(presidents), i.e. 4.
(fitSfx <- arima(presidents, order=c(2,0,1), seasonal=c(1,0,0),
                 fixed=c(NA, NA, 0.5, -0.1, 50), transform.pars=FALSE))
## The partly-fixed & smaller model seems better (as we "knew too much"):
AIC(fitSfx, arima(presidents, order=c(2,0,1), seasonal=c(1,0,0)))

## An example of ARIMA forecasting:
predict(fit3, 3)

```

arima.sim

```

arima.sim(model, n, rand.gen = rnorm, innov = rand.gen(n, ...),
          n.start = NA, start.innov = rand.gen(n.start, ...),
          ...)

```

```

model          armaorder
n

```

```

rand.gen
innov          rand.gen
n.start        NA
start.innov    n.startn.start
...           rand.genrnormsd

```

[arima](#)

```

ordermodelarimaorder
rand.gen

```

"ts"

[arima](#)

```

require(graphics)

arima.sim(n = 63, list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
          sd = sqrt(0.1796))
# mildly long-tailed
arima.sim(n = 63, list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
          rand.gen = function(n, ...) sqrt(0.1796) * rt(n, df = 5))

# An ARIMA simulation
ts.sim <- arima.sim(list(order = c(1,1,0), ar = 0.7), n = 200)
ts.plot(ts.sim)

```

arima0

[arima\(\)](#)

```

arima0(x, order = c(0, 0, 0),
       seasonal = list(order = c(0, 0, 0), period = NA),
       xreg = NULL, include.mean = TRUE, delta = 0.01,
       transform.pars = TRUE, fixed = NULL, init = NULL,
       method = c("ML", "CSS"), n.cond, optim.control = list())

## S3 method for class 'arima0'
predict(object, n.ahead = 1, newxreg, se.fit = TRUE, ...)

```

```

x
order          (p,d,q)
seasonal       frequency(x)listorderperiodorder
xreg           x
include.mean   TRUEFALSE
delta
transform.pars method = "CSS"
fixed          NAfixedtransform.pars = TRUE
init           fixed
method
n.cond
optim.control  optim
object        arima0
newxreg        xregn.ahead
n.ahead
se.fit
...

```

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + e_t + b_1 e_{t-1} + \cdots + b_q e_{t-q}$$

```
include.meanX - mX
```

```

optimxreg
predict.arima0

```

```

arima0"arima0"

coef
sigma2
var.coef      coef
loglik
arma
aic           method = "ML"
residuals
call
series        x
convergence    optim
n.cond
predict.arima0se.fit = TRUEpredse

```

```
deltadelta
transform.parstransform.parstransform.pars = FALSEtransform.pars = TRUE
delta
n.condn.condn.cond
```

```
arimaarima0()
```

```
arima.mlearima.mle
```

```
arimaartsdiag
```

```
## Not run: arima0(lh, order = c(1,0,0))
arima0(lh, order = c(3,0,0))
arima0(lh, order = c(1,0,1))
predict(arima0(lh, order = c(3,0,0)), n.ahead = 12)

arima0(lh, order = c(3,0,0), method = "CSS")

# for a model with as few years as this, we want full ML
(fit <- arima0(USAccDeaths, order = c(0,1,1),
              seasonal = list(order=c(0,1,1)), delta = -1))
predict(fit, n.ahead = 6)

arima0(LakeHuron, order = c(2,0,0), xreg = time(LakeHuron)-1920)
## Not run:
## presidents contains NAs
## graphs in example(acf) suggest order 1 or 3
(fit1 <- arima0(presidents, c(1, 0, 0), delta = -1)) # avoid warning
tsdiag(fit1)
(fit3 <- arima0(presidents, c(3, 0, 0), delta = -1)) # smaller AIC
tsdiag(fit3)
## End(Not run)
```

ARMAacf

```
ARMAacf(ar = numeric(), ma = numeric(), lag.max = r, pacf = FALSE)
```

ar

ma

lag.max max(p, q+1)p, q

pacf

$0, \dots, \max(p, q + 1)$

[armaARMatoMAacf2ARARMAacf](#)[filter](#)

```
ARMAacf(c(1.0, -0.25), 1.0, lag.max = 10)
```

```
## Example from Brockwell & Davis (1991, pp.92-4)
```

```
## answer:  $2^{(-n)} * (32/3 + 8 * n) / (32/3)$ 
```

```
n <- 1:10
```

```
a.n <-  $2^{(-n)} * (32/3 + 8 * n) / (32/3)$ 
```

```
(A.n <- ARMAacf(c(1.0, -0.25), 1.0, lag.max = 10))
```

```
stopifnot(all.equal(unname(A.n), c(1, a.n)))
```

```
ARMAacf(c(1.0, -0.25), 1.0, lag.max = 10, pacf = TRUE)
```

```
zapsmall(ARMAacf(c(1.0, -0.25), lag.max = 10, pacf = TRUE))
```

```
## Cov-Matrix of length-7 sub-sample of AR(1) example:
```

```
toeplitz(ARMAacf(0.8, lag.max = 7))
```

ARMAtoMA

```
ARMAtoMA(ar = numeric(), ma = numeric(), lag.max)
```

```
ar
ma
lag.max
```

[arimaARMAacf](#)

```
ARMAtoMA(c(1.0, -0.25), 1.0, 10)
## Example from Brockwell & Davis (1991, p.92)
## answer (1 + 3*n)*2^(-n)
n <- 1:10; (1 + 3*n)*2^(-n)
```

as.hclust	"hclust"
-----------	----------

```
"hclust"
```

```
as.hclust(x, ...)
```

```
x
...
```

```
"twins"dianaagnes"hclust"
```

```
"hclust"
```

hclustdianaagnes

```
x <- matrix(rnorm(30), ncol = 3)
hc <- hclust(dist(x), method = "complete")

if(require("cluster", quietly = TRUE)) {# is a recommended package
  ag <- agnes(x, method = "complete")
  hcag <- as.hclust(ag)
  ## The dendrograms order slightly differently:
  op <- par(mfrow = c(1,2))
  plot(hc) ; mtext("hclust", side = 1)
  plot(hcag); mtext("agnes", side = 1)
  detach("package:cluster")
}
```

asOneSidedFormula

asOneSidedFormula(object)

object

object

formula

```
(form <- asOneSidedFormula("age"))
stopifnot(exprs = {
  identical(form, asOneSidedFormula(form))
  identical(form, asOneSidedFormula(as.name("age")))
  identical(form, asOneSidedFormula(expression(age)))
})
asOneSidedFormula(quote(log(age)))
asOneSidedFormula(1)
```

ave

[xmean](#)

```
ave(x, ..., FUN = mean)
```

x

... [xinteraction](#)

FUN

```
yxyFUN(x)...g1, g2y[i]FUN(x[sub])subjg1[j] == g1[i]g2[j] == g2[i]
```

[meansplittapply](#)

```
require(graphics)
```

```
ave(1:3) # no grouping -> grand mean
```

```
attach(warpbreaks)
```

```
ave(breaks, wool)
```

```
ave(breaks, tension)
```

```
ave(breaks, tension, FUN = function(x) mean(x, trim = 0.1))
```

```
plot(breaks, main =
```

```
  "ave( Warpbreaks ) for  wool x  tension combinations")
```

```
lines(ave(breaks, wool, tension), type = "s", col = "blue")
```

```
lines(ave(breaks, wool, tension, FUN = median), type = "s", col = "green")
```

```
legend(40, 70, c("mean", "median"), lty = 1,
```

```
      col = c("blue", "green"), bg = "gray90")
```

```
detach()
```

```
# Running index per group
```

```
(g <- sample(c("u", "s", "e", "R"), 24, replace = TRUE))
```

```
ave(seq_along(g), g, FUN = seq_along)
```

bandwidth

[density](#)


```
bw.nrd0(x)
```

```
bw.nrd(x)
```

```
bw.ucv(x, nb = 1000, lower = 0.1 * hmax, upper = hmax,  
      tol = 0.1 * lower)
```

```
bw.bcv(x, nb = 1000, lower = 0.1 * hmax, upper = hmax,  
      tol = 0.1 * lower)
```

```
bw.SJ(x, nb = 1000, lower = 0.1 * hmax, upper = hmax,  
     method = c("ste", "dpi"), tol = 0.1 * lower)
```

```
x
```

```
nb
```

```
lowerupper      hmax
```

```
method          "ste""dpi"
```

```
tol             "ste"uniroot
```

```
bw.nrd0
```

```
bw.nrd
```

```
bw.ucvbw.bcv
```

```
bw.SJ
```

```
"ste"unirootc(lower, upper)
```

```
 $O(n^2)$ n = nb/2 $O(n)$ x
```

```
bwdensity
```

```
xdensity
```

```
density
```

```
bandwidth.nrducvbcvwidth.SJwidthdensity
```

```

require(graphics)

plot(density(precip, n = 1000))
rug(precip)
lines(density(precip, bw = "nrd"), col = 2)
lines(density(precip, bw = "ucv"), col = 3)
lines(density(precip, bw = "bcv"), col = 4)
lines(density(precip, bw = "SJ-ste"), col = 5)
lines(density(precip, bw = "SJ-dpi"), col = 6)
legend(55, 0.035,
      legend = c("nrd0", "nrd", "ucv", "bcv", "SJ-ste", "SJ-dpi"),
      col = 1:6, lty = 1)

```

bartlett.test

```

bartlett.test(x, ...)

## Default S3 method:
bartlett.test(x, g, ...)

## S3 method for class 'formula'
bartlett.test(formula, data, subset, na.action, ...)

x          "lm"
g          xx
formula    lhs ~ rhslhsrhs
data       model.frameformulaenvironment(formula)
subset
na.action  NAgetOption("na.action")
...

```

```

xgbartlett.test(x)bartlett.test(list(x, ...))
xgxx

```

```

"htest"

statistic
parameter
p.value
method      "Bartlett test of homogeneity of variances"
data.name

```

```
var.testfligner.testkansari.testmood.test
```

```
require(graphics)
```

```
plot(count ~ spray, data = InsectSprays)
bartlett.test(InsectSprays$count, InsectSprays$spray)
bartlett.test(count ~ spray, data = InsectSprays)
```

Beta

```
shape1shape2ncp
```

```
dbeta(x, shape1, shape2, ncp = 0, log = FALSE)
pbeta(q, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
qbeta(p, shape1, shape2, ncp = 0, lower.tail = TRUE, log.p = FALSE)
rbeta(n, shape1, shape2, ncp = 0)
```

```
xq
```

```
p
```

```
n          length(n) > 1
```

```
shape1shape2
```

```
ncp
```

```
loglog.p
```

```
lower.tail       $P[X \leq x]P[X > x]$ 
```

```
shape1= ashape2= b
```

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1} (1-x)^{b-1}$$

```
a > 0 b > 0 0 ≤ x ≤ 1 x = 0 x = 1
```

```
a/(a+b)ab/((a+b)2(a+b+1))a, b > 1= 1(a-1)/(a+b-2)ab[dpqr]beta()
```

```
pbeta
```

$$B_x(a,b) = \int_0^x t^{a-1} (1-t)^{b-1} dt,$$

```
Ix(a,b) = Bx(a,b)/B(a,b) B(a,b) = B1(a,b)beta
```

```
Ix(a,b)pbeta(x, a, b)
```

```
ncp= λX/(X+Y) X ~ χ22a(λ) Y ~ χ22b(λ) nλ
```

dbetapbetaqbetarbeta

NaN

nrbeta

n

ncp = 0ncpncp

dbetadb^{inom}

pbetalog_p = FALSE

log.p = TRUE-Inf $P = 0$ $P = 1$ -Inf^{warning}

pbeta

qbeta

rbeta

^{beta}

```
x <- seq(0, 1, length.out = 21)
```

```
dbeta(x, 1, 1)
```

```
pbeta(x, 1, 1)
```

```
## Visualization, including limit cases:
```

```
pl.beta <- function(a,b, asp = if(isLim) 1, ylim = if(isLim) c(0,1.1)) {
```

```
  if(isLim <- a == 0 || b == 0 || a == Inf || b == Inf) {
```

```
    eps <- 1e-10
```

```
    x <- c(0, eps, (1:7)/16, 1/2+c(-eps,0,eps), (9:15)/16, 1-eps, 1)
```

```
  } else {
```

```
    x <- seq(0, 1, length.out = 1025)
```

```
  }
```

```

fx <- cbind(dbeta(x, a,b), pbeta(x, a,b), qbeta(x, a,b))
f <- fx; f[fx == Inf] <- 1e100
matplot(x, f, ylab="", type="l", ylim=ylim, asp=asp,
        main = sprintf("[dpq]beta(x, a=%g, b=%g)", a,b))
abline(0,1, col="gray", lty=3)
abline(h = 0:1, col="gray", lty=3)
legend("top", paste0(c("d","p","q"), "beta(x, a,b)"),
       col=1:3, lty=1:3, bty = "n")
invisible(cbind(x, fx))
}
pl.beta(3,1)

pl.beta(2, 4)
pl.beta(3, 7)
pl.beta(3, 7, asp=1)

pl.beta(0, 0) ## point masses at {0, 1}

pl.beta(0, 2) ## point mass at 0 ; the same as
pl.beta(1, Inf)

pl.beta(Inf, 2) ## point mass at 1 ; the same as
pl.beta(3, 0)

pl.beta(Inf, Inf)# point mass at 1/2

```

binom.test

```

binom.test(x, n, p = 0.5,
           alternative = c("two.sided", "less", "greater"),
           conf.level = 0.95)

```

```

x
n          x
p
alternative "two.sided""greater""less"
conf.level

```

```

conf.level

```

```

"htest"

```

```

statistic
parameter

```

```

p.value
conf.int
estimate
null.value      p
alternative
method          "Exact binomial test"
data.name

```

`prop.test`

```

## Conover (1971), p. 97f.
## Under (the assumption of) simple Mendelian inheritance, a cross
## between plants of two particular genotypes produces progeny 1/4 of
## which are "dwarf" and 3/4 of which are "giant", respectively.
## In an experiment to determine if this assumption is reasonable, a
## cross results in progeny having 243 dwarf and 682 giant plants.
## If "giant" is taken as success, the null hypothesis is that p =
## 3/4 and the alternative that p != 3/4.
binom.test(c(682, 243), p = 3/4)
binom.test(682, 682 + 243, p = 3/4) # The same.
## => Data are in agreement with the null hypothesis.

```

Binomial

```

sizeprob
size

```

```

dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)

```

```

xq
p
n          length(n) > 1
size
prob
loglog.p
lower.tail   $P[X \leq x]P[X > x]$ 

```

size= nprob= p

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

$x = 0, \dots, n$ choose

xdbinom

$p(x)$

$x F(x) \geq p F$

dbinom pbinom qbinom rbinom

sizeNaN

nrbinom

n

dbinom

<https://www.r-project.org/doc/reports/CLoader-dbinom-2002.pdf>

pbinom beta

qbinom

rbinomsize < .Machine\$integer.max

dnbinom dpois

```
require(graphics)
# Compute P(45 < X < 55) for X Binomial(100,0.5)
sum(dbinom(46:54, 100, 0.5))

## Using "log = TRUE" for an extended range :
n <- 2000
k <- seq(0, n, by = 20)
plot (k, dbinom(k, n, pi/10, log = TRUE), type = "l", ylab = "log density",
      main = "dbinom(*, log=TRUE) is better than log(dbinom(*))")
lines(k, log(dbinom(k, n, pi/10)), col = "red", lwd = 2)
## extreme points are omitted since dbinom gives 0.
mtext("dbinom(k, log=TRUE)", adj = 0)
mtext("extended range", adj = 0, line = -1, font = 4)
mtext("log(dbinom(k))", col = "red", adj = 1)
```

biplot

```
biplot(x, ...)
```

```
## Default S3 method:
```

```
biplot(x, y, var.axes = TRUE, col, cex = rep(par("cex"), 2),  
       xlabs = NULL, ylabs = NULL, expand = 1,  
       xlim = NULL, ylim = NULL, arrow.len = 0.1,  
       main = NULL, sub = NULL, xlab = NULL, ylab = NULL, ...)
```

x	biplotbiplot.default
y	
var.axes	TRUE
col	palette
cex	
xlabs	x1:nNULL
ylabs	y1:nNULL
expand	
arrow.len	var.axesarrow.len = 0
xlimylim	
mainsubxlabylab...	

```
biplot.princompbiplot.default
```

```
biplotxlabsylabsxlabscex
```

```
biplot.princomp
```

`biplot.princomp`

`princomp`
`prcomp`

```
## S3 method for class 'prcomp'
biplot(x, choices = 1:2, scale = 1, pc.biplot = FALSE, ...)
```

```
## S3 method for class 'princomp'
biplot(x, choices = 1:2, scale = 1, pc.biplot = FALSE, ...)
```

```
x          "princomp"
choices
scale       $\lambda^{\text{scale}}$   $\lambda^{(1-\text{scale})}$  princomp  $0 \leq \text{scale} \leq 1$ 
          scale
pc.biplot   $\lambda = 1$ 
...        biplot.default
```

```
biplotpc.biplot = TRUE
```

`biplot`
`princomp`

```
require(graphics)
biplot(princomp(USArrests))
```

birthday

pbirthdayqbirthday

qbirthday(prob = 0.5, classes = 365, coincident = 2)
pbirthday(n, classes = 365, coincident = 2)

classes
prob
n
coincident

coincident > 2prob

qbirthday probkclasses
pbirthday

require(graphics)

the standard version
qbirthday() # 23
probability of > 2 people with the same birthday
pbirthday(23, coincident = 3)

examples from Diaconis & Mosteller p. 858.
'coincidence' is that husband, wife, daughter all born on the 16th
qbirthday(classes = 30, coincident = 3) # approximately 18
qbirthday(coincident = 4) # exact value 187
qbirthday(coincident = 10) # exact value 1181

same 4-digit PIN number
qbirthday(classes = 10^4)

0.9 probability of three or more coincident birthdays
qbirthday(coincident = 3, prob = 0.9)

Chance of 4 or more coincident birthdays in 150 people
pbirthday(150, coincident = 4)

100 or more coincident birthdays in 1000 people: very rare
pbirthday(1000, coincident = 100)

Box.test

```
Box.test(x, lag = 1, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

```
x
lag      lag
type
fitdf    x
```

```
ARMA(p, q)fitdf = p+qlag > fitdf
```

```
"htest"
statistic
parameter      fitdf
p.value
method
data.name
```

```
x <- rnorm(100)
Box.test(x, lag = 1)
Box.test(x, lag = 1, type = "Ljung")
```

"contrasts"

C(object, contr, how.many, ...)

object

contr contr.poly

how.many nlevels(object)

... contr

contr.treatment.helmert.sum.poly.contr.treatment

object"contrasts"

[contrastscontr.sum](#)

```
## reset contrasts to defaults
options(contrasts = c("contr.treatment", "contr.poly"))
tens <- with(warpbreaks, C(tension, poly, 1))
attributes(tens)
## tension SHOULD be an ordered factor, but as it is not we can use
aov(breaks ~ wool + tens + tension, data = warpbreaks)

## show the use of ... The default contrast is contr.treatment here
summary(lm(breaks ~ wool + C(tension, base = 2), data = warpbreaks))

# following on from help(esoph)
model3 <- glm(cbind(ncases, ncontrols) ~ agegp + C(tobgp, , 1) +
  C(alcgp, , 1), data = esoph, family = binomial())
summary(model3)
```

cancor

```
cancor(x, y, xcenter = TRUE, ycenter = TRUE)
```

x	$n \times p_1$
y	$n \times p_2$
xcenter	p_1 TRUE FALSE
ycenter	xcenter

yx

cor	
xcoef	x
ycoef	y
xcenter	x
ycenter	x

[qrsvd](#)

```
## signs of results are random
pop <- LifeCycleSavings[, 2:3]
oec <- LifeCycleSavings[, -(2:3)]
cancor(pop, oec)

x <- matrix(rnorm(150), 50, 3)
y <- matrix(rnorm(250), 50, 5)
(cxy <- cancor(x, y))
all(abs(cor(x %*% cxy$xcoef,
            y %*% cxy$ycoef)[,1:3] - diag(cxy $ cor)) < 1e-15)
all(abs(cor(x %*% cxy$xcoef) - diag(3)) < 1e-15)
all(abs(cor(y %*% cxy$ycoef) - diag(5)) < 1e-15)
```

case+variable.names

```
case.names(object, ...)
## S3 method for class 'lm'
case.names(object, full = FALSE, ...)

variable.names(object, ...)
## S3 method for class 'lm'
variable.names(object, full = FALSE, ...)
```

```
object
full          TRUE
...
```

[lmall.namesall.vars](#)

```
x <- 1:20
y <- setNames(x + (x/4 - 2)^3 + rnorm(20, sd = 3),
              paste("0", x, sep = "."))
ww <- rep(1, 20); ww[13] <- 0
summary(lmxy <- lm(y ~ x + I(x^2)+I(x^3) + I((x-10)^2), weights = ww),
        correlation = TRUE)
variable.names(lmxy)
variable.names(lmxy, full = TRUE) # includes the last
case.names(lmxy)
case.names(lmxy, full = TRUE)    # includes the 0-weight case
```

Cauchy

locationscale

```
dcauchy(x, location = 0, scale = 1, log = FALSE)
pcauchy(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qcauchy(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rcauchy(n, location = 0, scale = 1)
```

```
xq
p
n          length(n) > 1
locationscale
loglog.p
lower.tail       $P[X \leq x]P[X > x]$ 
```

```
locationscale01
```

```
ls
```

$$f(x) = \frac{1}{\pi s} \left(1 + \left(\frac{x-l}{s} \right)^2 \right)^{-1}$$

```
x
```

```
dcauchypcauchyqcauchyrcauchy
```

```
nrcauchy
```

```
n
```

```
dcauchypcauchyqcauchy
```

```
rcauchy
```

```
dt dcauchy(*, l = 0, s = 1)
```

```
dcauchy(-1:4)
```

```
chisq.test
```

```
chisq.test
```

```
chisq.test(x, y = NULL, correct = TRUE,
           p = rep(1/length(x), length(x)), rescale.p = FALSE,
           simulate.p.value = FALSE, B = 2000)
```

```

x          xy
y          xxy
correct     $|O - E|$ simulate.p.value = TRUE
p          xp
rescale.p  prescale.pp
simulate.p.value

```

B

```

xxyxxpp
xxy
simulate.p.valueFALSEcorrectTRUEBB = 20001/(B + 1)

```

```
pn = sum(x)
```

"htest"

```

statistic
parameter    NA
p.value
method
data.name
observed
expected
residuals    (observed - expected) / sqrt(expected)
stdres       (observed - expected) / sqrt(V) $V_{xn} * p * (1 - p)$ 

```

[ks.test](#)


```

## From Agresti(2007) p.39
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(gender = c("F", "M"),
                    party = c("Democrat", "Independent", "Republican"))
(Xsq <- chisq.test(M)) # Prints test summary
Xsq$observed # observed counts (same as M)
Xsq$expected # expected counts under the null
Xsq$residuals # Pearson residuals
Xsq$stdres # standardized residuals

## Effect of simulating p-values
x <- matrix(c(12, 5, 7, 7), ncol = 2)
chisq.test(x)$p.value # 0.4233
chisq.test(x, simulate.p.value = TRUE, B = 10000)$p.value
# around 0.29!

## Testing for population probabilities
## Case A. Tabulated data
x <- c(A = 20, B = 15, C = 25)
chisq.test(x)
chisq.test(as.table(x)) # the same
x <- c(89,37,30,28,2)
p <- c(40,20,20,15,5)
try(
  chisq.test(x, p = p) # gives an error
)
chisq.test(x, p = p, rescale.p = TRUE)
# works
p <- c(0.40,0.20,0.20,0.19,0.01)
# Expected count in category 5
# is 1.86 < 5 ==> chi square approx.
chisq.test(x, p = p) # maybe doubtful, but is ok!
chisq.test(x, p = p, simulate.p.value = TRUE)

## Case B. Raw data
x <- trunc(5 * runif(100))
chisq.test(table(x)) # NOT 'chisq.test(x)!'

```

Chisquare

χ^2 dfncp

```

dchisq(x, df, ncp = 0, log = FALSE)
pchisq(q, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)
qchisq(p, df, ncp = 0, lower.tail = TRUE, log.p = FALSE)
rchisq(n, df, ncp = 0)

```

`xq`
`p`
`n` `length(n) > 1`
`df`
`n`
`n`
`ncp`
`loglog.p`
`lower.tail` $P[X \leq x]P[X > x]$

`df=` $n \geq 0$

$$f_n(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}$$

$x > 0 f_0(x) := \lim_{n \rightarrow 0} f_n(x) = \delta_0(x) \delta$
 $n 2n$

`df=` $n ncp = \lambda$

$$f(x) = f_{n,\lambda}(x) = e^{-\lambda/2} \sum_{r=0}^{\infty} \frac{(\lambda/2)^r}{r!} f_{n+2r}(x)$$

$x \geq 0 n n \lambda$
 $E(X) = n + \lambda Var(X) = 2(n + 2 * \lambda) E((X - E(X))^3) = 8(n + 3 * \lambda)$
 $df = n n = 0 \lambda > 0 x = 0 pchisq(0, df=0, ncp=ncp) dchisq() df \rightarrow 0$
`ncppchisqqchisq`

`dchisqpchisqqchisqrchisq`
`NaN`
`nrchisq`
`n`

`ncp = 0 ncp ncp`
`ncp ncp`

`dchisqrchisq`
`pchisqncp < 80 ncp`

`qchisqpchisq`

$n\text{shape}\alpha = n/2\text{scale}\sigma = 2d\text{gamma}$

$\chi^2_2 = \text{Exp}(1/2)d\text{exp}$

```
require(graphics)

dchisq(1, df = 1:3)
pchisq(1, df = 3)
pchisq(1, df = 3, ncp = 0:4) # includes the above

x <- 1:10
## Chi-squared(df = 2) is a special exponential distribution
all.equal(dchisq(x, df = 2), dexp(x, 1/2))
all.equal(pchisq(x, df = 2), pexp(x, 1/2))

## non-central RNG -- df = 0 with ncp > 0: Z0 has point mass at 0!
Z0 <- rchisq(100, df = 0, ncp = 2.)
graphics::stem(Z0)

## visual testing
## do P-P plots for 1000 points at various degrees of freedom
L <- 1.2; n <- 1000; pp <- ppoints(n)
op <- par(mfrow = c(3,3), mar = c(3,3,1,1)+.1, mgp = c(1.5,.6,0),
          oma = c(0,0,3,0))
for(df in 2^(4*rnorm(9))) {
  plot(pp, sort(pchisq(rr <- rchisq(n, df = df, ncp = L), df = df, ncp = L)),
       ylab = "pchisq(rchisq(.),.)", pch = ".")
  mtext(paste("df = ", formatC(df, digits = 4)), line = -2, adj = 0.05)
  abline(0, 1, col = 2)
}
mtext(expression("P-P plots : Noncentral " *
                 chi^2 * "(n=1000, df=X, ncp= 1.2)"),
       cex = 1.5, font = 2, outer = TRUE)
par(op)

## "analytical" test
lam <- seq(0, 100, by = .25)
p00 <- pchisq(0, df = 0, ncp = lam)
p.0 <- pchisq(1e-300, df = 0, ncp = lam)
stopifnot(all.equal(p00, exp(-lam/2)),
          all.equal(p.0, exp(-lam/2)))
```

cmdscale

```
cmdscale(d, k = 2, eig = FALSE, add = FALSE, x.ret = FALSE,
         list. = eig || add || x.ret)
```

```

d          dist
k          {1, 2, \dots, n - 1}
eig
add        c*
x.ret
list.      list n \times k

```

```

nn - 1cmdscalekkk
cmdscaleprcomp
add = TRUEc*d_{ij} + c*n - 1n - 1

```

```

list.k
list
points      k
eig          neigkn - 1
x            x.ret
ac           c*\emptysetadd = FALSE
GOF          (g_1, g_2)g_i = (\sum_{j=1}^k \lambda_j)/(\sum_{j=1}^n T_i(\lambda_j))\lambda_j T_1(v) = |v|T_2(v) = max(v, 0)

```

```

dist
isoMDSsammon

```

```

require(graphics)

loc <- cmdscale(eurodist)
x <- loc[, 1]
y <- -loc[, 2] # reflect so North is at the top
## note asp = 1, to ensure Euclidean distances are represented correctly
plot(x, y, type = "n", xlab = "", ylab = "", asp = 1, axes = FALSE,
      main = "cmdscale(eurodist)")
text(x, y, rownames(loc), cex = 0.6)

```

coef

coefcoefficients

```
coef(object, ...)
coefficients(object, ...)
## Default S3 method:
coef(object, complete = TRUE, ...)
## S3 method for class 'aov'
coef(object, complete = FALSE, ...)
```

object

complete lmaovNAaliaslm()aov()

...

coefcoefcoefficients

"aov"aliascomplete = FALSE

completevcovcoefaovcomplete = *p <- length(coef(obj, complete = TF))dim(vcov(obj,
complete = TF)) == c(p,p)complete

object

"maov"aov

fitted.valuesresidualsglm

x <- 1:5; coef(lm(c(1:3, 7, 6) ~ x))

complete.cases

complete.cases(...)

...

[lengthis.na"POSIXlt"](#)

[is.nana.omitna.fail](#)

```
x <- airquality[, -1] # x is a regression design matrix
y <- airquality[, 1] # y is the corresponding response
```

```
stopifnot(complete.cases(y) != is.na(y))
ok <- complete.cases(x, y)
sum(!ok) # how many are not "ok" ?
x <- x[ok,]
y <- y[ok]
```

confint

["lm"](#)

```
confint(object, parm, level = 0.95, ...)
## Default S3 method:
confint(object, parm, level = 0.95, ...)
## S3 method for class 'lm'
confint(object, parm, level = 0.95, ...)
## S3 method for class 'glm'
confint(object, parm, level = 0.95, trace = FALSE, test=c("LRT", "Rao"), ...)
## S3 method for class 'nls'
confint(object, parm, level = 0.95, ...)
```

object

parm

level

trace

test

...

confintcoefvcov

"lm"*t*

"glm""nls"

confint.glmconfint.nls

```
fit <- lm(100/mpg ~ disp + hp + wt + am, data = mtcars)
confint(fit)
confint(fit, "wt")

## from example(glm)
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3, 1, 9); treatment <- gl(3, 3)
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
confint(glm.D93)
confint.default(glm.D93) # based on asymptotic normality
```

constrOptim

```
constrOptim(theta, f, grad, ui, ci, mu = 1e-04, control = list(),
             method = if(is.null(grad)) "Nelder-Mead" else "BFGS",
             outer.iterations = 100, outer.eps = 1e-05, ...,
             hessian = FALSE)
```

```

theta           $p$ 
f
grad           functionNULL
ui              $k \times p$ 
ci              $k$ 
mu
controlmethodhessian
              optim
outer.iterations

outer.eps
...           fgradoptim

ui %%% theta - ci >= 0
optim
mumu
optim
f...optimoptimoptimcontrol$fnscale
gradmethod = "Nelder-Mead"

optimbarrier.valueouter.iterationsoptimcountsoptim()$counts

optimmethod = "L-BFGS-B"

## from optim
fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
grr <- function(x) { ## Gradient of 'fr'
  x1 <- x[1]
  x2 <- x[2]
  c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
    200 * (x2 - x1 * x1))
}

optim(c(-1.2,1), fr, grr)
#Box-constraint, optimum on the boundary
constrOptim(c(-1.2,0.9), fr, grr, ui = rbind(c(-1,0), c(0,-1)), ci = c(-1,-1))

```



```

# x <= 0.9, y - x > 0.1
constrOptim(c(.5,0), fr, grr, ui = rbind(c(-1,0), c(1,-1)), ci = c(-0.9,0.1))

## Solves linear and quadratic programming problems
## but needs a feasible starting value
#
# from example(solve.QP) in 'quadprog'
# no derivative
fQP <- function(b) {-sum(c(0,5,0)*b)+0.5*sum(b*b)}
Amat <- matrix(c(-4,-3,0,2,1,0,0,-2,1), 3, 3)
bvec <- c(-8, 2, 0)
constrOptim(c(2,-1,-1), fQP, NULL, ui = t(Amat), ci = bvec)
# derivative
gQP <- function(b) {-c(0, 5, 0) + b}
constrOptim(c(2,-1,-1), fQP, gQP, ui = t(Amat), ci = bvec)

## Now with maximisation instead of minimisation
hQP <- function(b) {sum(c(0,5,0)*b)-0.5*sum(b*b)}
constrOptim(c(2,-1,-1), hQP, NULL, ui = t(Amat), ci = bvec,
            control = list(fnscale = -1))

```

contrast

```

contr.helmert(n, contrasts = TRUE, sparse = FALSE)
contr.poly(n, scores = 1:n, contrasts = TRUE, sparse = FALSE)
contr.sum(n, contrasts = TRUE, sparse = FALSE)
contr.treatment(n, base = 1, contrasts = TRUE, sparse = FALSE)
contr.SAS(n, contrasts = TRUE, sparse = FALSE)

```

```

n
contrasts
sparse      dgCMatrix
scores
base        contrastsFALSE

```

```

ncontrastsFALSEcontr.polycontrasts = FALSE
contr.helmertcontr.polycontr.sum
contr.treatmentbase
contr.SAScontr.treatment
sparsesparse = TRUEcontr.polycontr.helmert

```

```

nkk=n-1contrastsTRUEk=ncontrastsFALSE

```

contrastsCaovglm

```
(cH <- contr.helmert(4))
apply(cH, 2, sum) # column sums are 0
crossprod(cH) # diagonal -- columns are orthogonal
contr.helmert(4, contrasts = FALSE) # just the 4 x 4 identity matrix

(cT <- contr.treatment(5))
all(crossprod(cT) == diag(4)) # TRUE: even orthonormal

(cT. <- contr.SAS(5))
all(crossprod(cT.) == diag(4)) # TRUE

zapsmall(cP <- contr.poly(3)) # Linear and Quadratic
zapsmall(crossprod(cP), digits = 15) # orthonormal up to fuzz
```

contrasts

```
contrasts(x, contrasts = TRUE, sparse = FALSE)
contrasts(x, how.many = NULL) <- value
```

```
x
contrasts
sparse      dgCMatrix
how.many    xvalue
value       dMatrixx
```

```
options("contrasts")
xc(FALSE, TRUE)
contrastsxcontrastscontrasts = TRUEcontr.treatmentcontrasts = FALSEcontrasts
contrasts = TRUEsparse
valuehow.manyhow.manyvalue
```

[Ccontr.helmertcontr.polycontr.sumcontr.treatmentglmavlm](#)

```
utils::example(factor)
fff <- ff[, drop = TRUE] # reduce to 5 levels.
contrasts(fff) # treatment contrasts by default
contrasts(C(fff, sum))
contrasts(fff, contrasts = FALSE) # the 5x5 identity matrix

contrasts(fff) <- contr.sum(5); contrasts(fff) # set sum contrasts
contrasts(fff, 2) <- contr.sum(5); contrasts(fff) # set 2 contrasts
# supply 2 contrasts, compute 2 more to make full set of 4.
contrasts(fff) <- contr.sum(5)[, 1:2]; contrasts(fff)

## using sparse contrasts: % useful, once model.matrix() works with these :
ffs <- fff
contrasts(ffs) <- contr.sum(5, sparse = TRUE)[, 1:2]; contrasts(ffs)
stopifnot(all.equal(ffs, fff))
contrasts(ffs) <- contr.sum(5, sparse = TRUE); contrasts(ffs)
```

convolve

```
convolve(x, y, conj = TRUE, type = c("circular", "open", "filter"))
```

```
xy
conj          TRUE
type          "circular" "open" "filter" "circular"
              "open" "filter" "0" "filter" "open" xy
```

[fft](#)

```
xycircular
xyconvolve(x, rev(y), type = "o")
```

```
r <- convolve(x, y, type = "open") n <- length(x) m <- length(y)
```

$$r_k = \sum_i x_{k-m+i} y_i$$

$ik = 1, \dots, n + m - 1$

type == "circular" $n = mi, k = 1, \dots, nx_j := x_{n+j} j < 1$

fftnextnfilter

```
require(graphics)

x <- c(0,0,0,100,0,0,0)
y <- c(0,0,1, 2 ,1,0,0)/4
zapsmall(convolve(x, y))          # *NOT* what you first thought.
zapsmall(convolve(x, y[3:5], type = "f")) # rather
x <- rnorm(50)
y <- rnorm(50)
# Circular convolution *has* this symmetry:
all.equal(convolve(x, y, conj = FALSE), rev(convolve(rev(y),x)))

n <- length(x <- -20:24)
y <- (x-10)^2/1000 + rnorm(x)/8

Han <- function(y) # Hanning
  convolve(y, c(1,2,1)/4, type = "filter")

plot(x, y, main = "Using convolve(.) for Hanning filters")
lines(x[-c(1 , n)      ], Han(y), col = "red")
lines(x[-c(1:2, (n-1):n)], Han(Han(y)), lwd = 2, col = "dark blue")
```

cophenetic

```
cophenetic(x)
## Default S3 method:
cophenetic(x)
## S3 method for class 'dendrogram'
cophenetic(x)
```

```
x          "hclust"as.hclust()"agnes"
```

```
copheneticas.hclust()cophenetic()
"dendrogram"
```

"dist"

disthclust

```
require(graphics)

d1 <- dist(USArrests)
hc <- hclust(d1, "ave")
d2 <- cophenetic(hc)
cor(d1, d2) # 0.7659

## Example from Sneath & Sokal, Fig. 5-29, p.279
d0 <- c(1,3.8,4.4,5.1, 4,4.2,5, 2.6,5.3, 5.4)
attributes(d0) <- list(Size = 5, diag = TRUE)
class(d0) <- "dist"
names(d0) <- letters[1:5]
d0
utils::str(upgma <- hclust(d0, method = "average"))
plot(upgma, hang = -1)
#
(d.coph <- cophenetic(upgma))
cor(d0, d.coph) # 0.9911
```

cor

varcovcorxxyxyxy

cov2cor

var(x, y = NULL, na.rm = FALSE, use)

cov(x, y = NULL, use = "everything",
method = c("pearson", "kendall", "spearman"))

cor(x, y = NULL, use = "everything",
method = c("pearson", "kendall", "spearman"))

cov2cor(V)

```

x
y          NULLxy = x
na.rm
use        "everything""all.obs""complete.obs""na.or.complete"
          "pairwise.complete.obs"
method     "pearson""kendall""spearman"
v

covcorxxy

is.numeric"kendall""spearman"xtfrm

varcovna.rmusena.rmTRUEuse = "na.or.complete"use = "everything"

use"everything"NA
use"all.obs"use"complete.obs"
"na.or.complete"NAuse"pairwise.complete.obs"NAcovvar"pairwise.complete.obs"
"pearson"var(double(0), use = *)NAuse = "everything""na.or.complete"

n - 1NA

cor()method"kendall""spearman" $\tau\rho$ 
cov()"spearman"cor(R(x), R(y))cov(., .)R(u) := rank(u, na.last = "keep")use

 $\tau_b$ 

sweep(., FUN = "/" )cov2cor

r <- cor(*, use = "all.obs")all(abs(r) <= 1)

cor.fk

cor.test
cov.wt
sd

```

```

var(1:10) # 9.166667

var(1:5, 1:5) # 2.5

## Two simple vectors
cor(1:10, 2:11) # == 1

## Correlation Matrix of Multivariate sample:
(C1 <- cor(longley))
## Graphical Correlation Matrix:
symnum(C1) # highly correlated

## Spearman's rho and Kendall's tau
symnum(c1S <- cor(longley, method = "spearman"))
symnum(c1K <- cor(longley, method = "kendall"))
## How much do they differ?
i <- lower.tri(C1)
cor(cbind(P = C1[i], S = c1S[i], K = c1K[i]))

## cov2cor() scales a covariance matrix by its diagonal
##           to become the correlation matrix.
cov2cor # see the function definition {and learn ..}
stopifnot(all.equal(C1, cov2cor(cov(longley))),
           all.equal(cor(longley, method = "kendall"),
                     cov2cor(cov(longley, method = "kendall"))))

##--- Missing value treatment:
C1 <- cov(swiss)
range(eigen(C1, only.values = TRUE)$values) # 6.19      1921

## swM := "swiss" with 3 "missing"s :
swM <- swiss
colnames(swM) <- abbreviate(colnames(swiss), minlength=6)
swM[1,2] <- swM[7,3] <- swM[25,5] <- NA # create 3 "missing"

## Consider all 5 "use" cases :
(C <- cov(swM)) # use="everything" quite a few NA's in cov.matrix
try(cov(swM, use = "all")) # Error: missing obs...
C2 <- cov(swM, use = "complete")
stopifnot(identical(C2, cov(swM, use = "na.or.complete")))
range(eigen(C2, only.values = TRUE)$values) # 6.46      1930
C3 <- cov(swM, use = "pairwise")
range(eigen(C3, only.values = TRUE)$values) # 6.19      1938

## Kendall's tau doesn't change much:
symnum(Rc <- cor(swM, method = "kendall", use = "complete"))
symnum(Rp <- cor(swM, method = "kendall", use = "pairwise"))
symnum(R <- cor(swiss, method = "kendall"))

## "pairwise" is closer componentwise,
summary(abs(c(1 - Rp/R.)))
summary(abs(c(1 - Rc/R.)))

## but "complete" is closer in Eigen space:
EV <- function(m) eigen(m, only.values=TRUE)$values

```

```
summary(abs(1 - EV(Rp)/EV(R.)) / abs(1 - EV(Rc)/EV(R.)))
```

`cor.test`

$\tau\rho$

```
cor.test(x, ...)
```

```
## Default S3 method:
```

```
cor.test(x, y,
         alternative = c("two.sided", "less", "greater"),
         method = c("pearson", "kendall", "spearman"),
         exact = NULL, conf.level = 0.95, continuity = FALSE, ...)
```

```
## S3 method for class 'formula'
```

```
cor.test(formula, data, subset, na.action, ...)
```

<code>xy</code>	<code>xy</code>
<code>alternative</code>	<code>"two.sided""greater""less""greater""less"</code>
<code>method</code>	<code>"pearson""kendall""spearman"</code>
<code>exact</code>	<code>$\tau\rho$NULL</code>
<code>conf.level</code>	
<code>continuity</code>	<code>$\tau\rho$</code>
<code>formula</code>	<code>~ u + vuv</code>
<code>data</code>	<code>model.frameformulaenvironment(formula)</code>
<code>subset</code>	
<code>na.action</code>	<code>NAgetOption("na.action")</code>
<code>...</code>	

`[-1,1]0`

`method"pearson"cor(x, y)tlength(x)-2`

`method"kendall""spearman" $\tau\rho$`

`exact`

`$n < 1290$ exact = TRUE $tn < 10$`


```

"htest"
statistic
parameter      t
p.value
estimate       "cor""tau""rho"
null.value     0
alternative
method
data.name
conf.int

```

ρ

[Kendall](#)

[pKendallpSpearman](#)[spearman.test](#)

```

## Hollander & Wolfe (1973), p. 187f.
## Assessment of tuna quality. We compare the Hunter L measure of
## lightness to the averages of consumer panel scores (recoded as
## integer values from 1 to 6 and averaged over 80 such values) in
## 9 lots of canned tuna.

x <- c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1)
y <- c( 2.6,  3.1,  2.5,  5.0,  3.6,  4.0,  5.2,  2.8,  3.8)

## The alternative hypothesis of interest is that the
## Hunter L value is positively associated with the panel score.

cor.test(x, y, method = "kendall", alternative = "greater")
## => p=0.05972

cor.test(x, y, method = "kendall", alternative = "greater",
         exact = FALSE) # using large sample approximation
## => p=0.04765

## Compare this to
cor.test(x, y, method = "spearman", alternative = "g")
cor.test(x, y,                alternative = "g")

## Formula interface.
require(graphics)
pairs(USJudgeRatings)
cor.test(~ CONT + INTG, data = USJudgeRatings)

```

cov.wt

```
cov.wt(x, wt = rep(1/nrow(x), nrow(x)), cor = FALSE, center = TRUE,  
       method = c("unbiased", "ML"))
```

```
x  
wt          x  
cor  
center      TRUEFALSEcenterx  
method
```

```
method = "unbiased" $1/n(n-1)$ 
```

```
cov  
center  
n.obs      x  
wt  
cor        corTRUE
```

[covvar](#)

```
(xy <- cbind(x = 1:10, y = c(1:3, 8:5, 8:10)))  
w1 <- c(0,0,0,1,1,1,1,1,0,0)  
cov.wt(xy, wt = w1) # i.e. method = "unbiased"  
cov.wt(xy, wt = w1, method = "ML", cor = TRUE)
```

cpgram

```
cpgram(ts, taper = 0.1,  
      main = paste("Series: ", deparse1(substitute(ts))),  
      ci.col = "blue")
```

ts

taper

main

ci.col

```
require(graphics)
```

```
par(pty = "s", mfrow = c(1,2))
```

```
cpgram(lh)
```

```
lh.ar <- ar(lh, order.max = 9)
```

```
cpgram(lh.ar$resid, main = "AR(3) fit to lh")
```

```
cpgram(ldeaths)
```

cutree

[hclust](#)

```
cutree(tree, k = NULL, h = NULL)
```

```
tree          hclustcutree()mergeheightlabels  
k  
h  
khkh
```

cutreekhkh

[hclustdendrogram](#)

```
hc <- hclust(dist(USArrests))  
  
cutree(hc, k = 1:5) #k = 1 is trivial  
cutree(hc, h = 250)  
  
## Compare the 2 and 4 grouping:  
g24 <- cutree(hc, k = c(2,4))  
table(grp2 = g24[, "2"], grp4 = g24[, "4"])
```

decompose

```
decompose(x, type = c("additive", "multiplicative"), filter = NULL)
```

```
x
type
filter          NULL
```

$$Y_t = T_t + S_t + e_t$$

$$Y_t = T_t S_t e_t$$

```
filterNULL
```

```
x
```

```
"decomposed.ts"
```

```
x
seasonal
figure
trend
random
type          type
```

```
stl
```

```
<David.Meyer@wu.ac.at>
```

```
stl
```

```
require(graphics)
```

```
m <- decompose(co2)
m$figure
plot(m)
```

```
## example taken from Kendall/Stuart
x <- c(-50, 175, 149, 214, 247, 237, 225, 329, 729, 809,
      530, 489, 540, 457, 195, 176, 337, 239, 128, 102, 232, 429, 3,
      98, 43, -141, -77, -13, 125, 361, -45, 184)
x <- ts(x, start = c(1951, 1), end = c(1958, 4), frequency = 4)
m <- decompose(x)
## seasonal figure: 6.25, 8.62, -8.84, -6.03
round(decompose(x)$figure / 10, 2)
```

delete.response

```
delete.responseterms
drop.terms[.terms
reformulatelength(termlabels) > 1+make.namesparseresponse
```

```
delete.response(termobj)

reformulate(termlabels, response = NULL, intercept = TRUE, env = parent.frame())

drop.terms(termobj, dropx = NULL, keep.response = FALSE)

## S3 method for class 'terms'
termobj[i]
```

```
termobj          terms
termlabels
response         NULL
intercept
env              environmentformula
dropxi           labels(termobj)"term.labels"termobjdropxidropx=NULL
keep.response
```

```
delete.responstedrop.termsterms
reformulateformula
```

terms

```
ff <- y ~ z + x + w
tt <- terms(ff)
tt
delete.response(tt)
drop.terms(tt, 2:3, keep.response = TRUE)
tt[-1]
tt[2:3]
reformulate(attr(tt, "term.labels"))

## keep LHS :
reformulate("x*w", ff[[2]])
fS <- surv(ft, case) ~ a + b
reformulate(c("a", "b*f"), fS[[2]])
```

```
## using non-syntactic names:
reformulate(c("`P/E`", "`% Growth`"), response = as.name("+--"))

x <- c("a name", "another name")
tryCatch( reformulate(x), error = function(e) "Syntax error." )
## rather backquote the strings in x :
reformulate(sprintf("`%s`", x))

stopifnot(identical(      ~ var, reformulate("var")),
           identical(~ a + b + c, reformulate(letters[1:3])),
           identical( y ~ a + b, reformulate(letters[1:2], "y")))
)
```

dendrapply

```
FUNdendrogramy <- dendrapply(x, fn)yxy.node[j] <- FUN( x.node[j], ...)y.node[j]
```

```
dendrapply(X, FUN, ...)
```

```
X           "dendrogram"
FUN         attributes
...         FUN
```

```
XFUN <- function(X) { attributes(X) <- .....; X }
```

[dendrogram](#)

[as.dendrogram](#)[lapply](#)[listrapply](#)

```
require(graphics)
```

```
## a smallish simple dendrogram
dhc <- as.dendrogram(hc <- hclust(dist(USArrests), "ave"))
(dhc21 <- dhc[[2]][[1]])
```

```
## too simple:
dendrapply(dhc21, function(n) utils::str(attributes(n)))
```

```
## toy example to set colored leaf labels :
```

```

local({
  collab <- function(n) {
    if(is.leaf(n)) {
      a <- attributes(n)
      i <- i+1
      attr(n, "nodePar") <-
        c(a$nodePar, list(lab.col = mycols[i], lab.font = i%3))
    }
    n
  }
  mycols <- grDevices::rainbow(attr(dhc21, "members"))
  i <- 0
})
dL <- dendrapply(dhc21, collab)
op <- par(mfrow = 2:1)
plot(dhc21)
plot(dL) ## --> colored labels!
par(op)

```

dendrogram

"dendrogram"

```

as.dendrogram(object, ...)
## S3 method for class 'hclust'
as.dendrogram(object, hang = -1, check = TRUE, ...)

## S3 method for class 'dendrogram'
as.hclust(x, ...)

## S3 method for class 'dendrogram'
plot(x, type = c("rectangle", "triangle"),
      center = FALSE,
      edge.root = is.leaf(x) || !is.null(attr(x, "edgetext")),
      nodePar = NULL, edgePar = list(),
      leaflab = c("perpendicular", "textlike", "none"),
      dLeaf = NULL, xlab = "", ylab = "", xaxt = "n", yaxt = "s",
      horiz = FALSE, frame.plot = FALSE, xlim, ylim, ...)

## S3 method for class 'dendrogram'
cut(x, h, ...)

## S3 method for class 'dendrogram'
merge(x, y, ..., height,
      adjust = c("auto", "add.max", "none"))

## S3 method for class 'dendrogram'
nobs(object, ...)

## S3 method for class 'dendrogram'

```



```
print(x, digits, ...)
```

```
## S3 method for class 'dendrogram'  
rev(x)
```

```
## S3 method for class 'dendrogram'  
str(object, max.level = NA, digits.d = 3,  
     give.attr = FALSE, wid = getOption("width"),  
     nest.lev = 0, indent.str = "",  
     last.str = getOption("str.dendrogram.last"), stem = "--",  
     ...)
```

```
is.leaf(object)
```

```
object      "dendrogram"  
xy          "dendrogram"  
hang        plot.hclust  
check       objectxhclust()check=TRUE  
type  
center      TRUE  
edge.root  
nodePar      listpointsNULLpchcexcolxpdbgpch1:2pch = NAnodePar  
edgePar      listsegmentsedgetextcolltylwdp.colp.lwdp.ltypolygonl.colnodePar  
leaflab      "perpendicular"  
             "textlike"  
             "none"  
dleaf        NULL  
horiz  
frame.plot   plot.default  
h  
height       NULL  
adjust       "auto"1"add.max"adjust  
xlimylim     plot.default  
...xlabylabxaxyaxt
```

```
digits       print.default  
max.leveldigits.dgive.attrwidnest.levindent.str  
             strstr.default()give.attr = FALSEheightmembers  
last.strstem str()last.str = ""last.str = "~"
```

```
zz[[1]]z[[1]][[2]]z[[c(1,2)]]membersheightleaf
```

```
members
```

```
height
```

```

midpoint plot(*, center = FALSE)
label
x.member cut()$uppermembershoriz = FALSE
edgetext
nodePar pointsnodePar
edgePar segmentsedgetextedgePar
leaf TRUE

cut.dendrogram()$upper$lowerdendrogramdendrogram
[[printstr"dendrogram"[
"hclust""dendrogram"as.dendrogram()as.hclust()
rev.dendrogramxreorder.dendrogram
merge(x, y, ...)xyadjust = "none"as.dendrogram(hclust(..))
nobs(object)members
is.leaf(object)object
plotNode()plotNodeLimit()

```

```

merge()options("expressions")Cstack_info()[["size"]]

```

```

plot() type = "triangle"center = TRUE
str(d) str(unclass(d))

```

```

dendrapplyorder.dendrogramreorder.dendrogramlabels

```

```

require(graphics); require(utils)

hc <- hclust(dist(USArrests), "ave")
(dend1 <- as.dendrogram(hc)) # "print()" method
str(dend1) # "str()" method
str(dend1, max.level = 2, last.str = "'') # only the first two sub-levels
oo <- options(str.dendrogram.last = "\\") # yet another possibility
str(dend1, max.level = 2) # only the first two sub-levels
options(oo) # .. resetting them

op <- par(mfrow = c(2,2), mar = c(5,2,1,4))
plot(dend1)
## "triangle" type and show inner nodes:
plot(dend1, nodePar = list(pch = c(1,NA), cex = 0.8, lab.cex = 0.8),
     type = "t", center = TRUE)
plot(dend1, edgePar = list(col = 1:2, lty = 2:3),
     dLeaf = 1, edge.root = TRUE)
plot(dend1, nodePar = list(pch = 2:1, cex = .4*2:1, col = 2:3),
     horiz = TRUE)

## simple test for as.hclust() as the inverse of as.dendrogram():

```

```

stopifnot(identical(as.hclust(dend1)[1:4], hc[1:4]))

dend2 <- cut(dend1, h = 70)
## leaves are wrong horizontally in R 4.0 and earlier:
plot(dend2$upper)
plot(dend2$upper, nodePar = list(pch = c(1,7), col = 2:1))
## dend2$lower is *NOT* a dendrogram, but a list of .. :
plot(dend2$lower[[3]], nodePar = list(col = 4), horiz = TRUE, type = "tr")
## "inner" and "leaf" edges in different type & color :
plot(dend2$lower[[2]], nodePar = list(col = 1), # non empty list
     edgePar = list(lty = 1:2, col = 2:1), edge.root = TRUE)
par(op)
d3 <- dend2$lower[[2]][[2]][[1]]
stopifnot(identical(d3, dend2$lower[[2]][[c(2,1)]]))
str(d3, last.str = "")

## to peek at the inner structure "if you must", use '[' indexing :
str(d3[2][[1]]) ## or the full
str(d3[])

## merge() to join dendrograms:
(d13 <- merge(dend2$lower[[1]], dend2$lower[[3]]))
## merge() all parts back (using default 'height' instead of original one):
den.1 <- Reduce(merge, dend2$lower)
## or merge() all four parts at same height --> 4 branches (!)
d. <- merge(dend2$lower[[1]], dend2$lower[[2]], dend2$lower[[3]],
            dend2$lower[[4]])
## (with a warning) or the same using do.call :
stopifnot(identical(d., do.call(merge, dend2$lower)))
plot(d., main = "merge(d1, d2, d3, d4) |-> dendrogram with a 4-split")

## "Zoom" in to the first dendrogram :
plot(dend1, xlim = c(1,20), ylim = c(1,50))

nP <- list(col = 3:2, cex = c(2.0, 0.75), pch = 21:22,
           bg = c("light blue", "pink"),
           lab.cex = 0.75, lab.col = "tomato")
plot(d3, nodePar = nP, edgePar = list(col = "gray", lwd = 2), horiz = TRUE)
addE <- function(n) {
  if(!is.leaf(n)) {
    attr(n, "edgePar") <- list(p.col = "plum")
    attr(n, "edgetext") <- paste(attr(n, "members"), "members")
  }
  n
}
d3e <- dendrapply(d3, addE)
plot(d3e, nodePar = nP)
plot(d3e, nodePar = nP, leaflab = "textlike")

```

density

density

```

density(x, ...)
## Default S3 method:
density(x, bw = "nrd0", adjust = 1,
        kernel = c("gaussian", "epanechnikov", "rectangular",
                    "triangular", "biweight",
                    "cosine", "optcosine"),
        weights = NULL, window = kernel, width,
        give.Rkern = FALSE, subdensity = FALSE,
        warnWbw = var(weights) > 0,
        n = 512, from, to, cut = 3, ext = 4,
        old.coords = FALSE,
        na.rm = FALSE, ...)

```

x	
bw	<code>bw</code> <code>bw.nrd</code> <code>"nrd0"</code> <code>"SJ"</code> <code>bw.adjust</code>
adjust	<code>adjust*bw</code>
kernelwindow	<code>"gaussian"</code> <code>"rectangular"</code> <code>"triangular"</code> <code>"epanechnikov"</code> <code>"biweight"</code> <code>"cosine"</code> <code>"optcosine"</code> <code>"gaussian"</code> <code>"cosine"</code> <code>"optcosine"</code> <code>"cosine"</code>
weights	<code>x=NULL</code> <code>weights = rep(1/nx, nx)</code> <code>nxx[]</code> <code>na.rm = TRUE</code> <code>NA</code> <code>x</code> <code>bw.cdensity(x = rep(x, cn))</code> <code>weights</code>
width	<code>bw</code> <code>bw.width</code> <code>width</code>
give.Rkern	<code>kernel</code>
subdensity	<code>weights</code> <code>warning</code>
warnWbw	<code>logical</code> <code>weights</code> <code>bw</code> <code>character</code> <code>warning</code>
n	<code>n > 512</code> <code>fft</code> <code>approx</code> <code>n</code>
fromto	<code>cut * bw</code> <code>range(x)</code>
cut	<code>from</code> <code>to</code> <code>cut</code>
ext	<code>4</code> <code>from</code> <code>to</code> <code>lo <- from - ext * bw</code> <code>up <- to + ext * bw</code> <code>n</code>
old.coords	<code>logical</code> <code>(1 + 1/(2n - 2))</code>
na.rm	<code>TRUE</code> <code>x</code> <code>FALSE</code>
...	

`density.default`

$\sigma_K^2 = \int t^2 K(t) dt = 1/bw R(K) = \int K^2(t) dt$
 $\sigma_K R(K) R(K)$
`give.Rkern = TRUE`
`x +/- Inf (-Inf, +Inf)`

```
give.Rkern $R(K)$ "density"
```

```
x          n
```

```
y
```

```
bw
```

```
n
```

```
call
```

```
data.name      x
```

```
has.na         FALSE
```

```
printsummaryxy
```

```
bw.nrdplot.densityhistfftconvolve
```

```
require(graphics)
```

```
plot(density(c(-20, rep(0,98), 20)), xlim = c(-4, 4)) # IQR = 0
```

```
# The Old Faithful geyser data
```

```
d <- density(faithful$eruptions, bw = "sj")
```

```
d
```

```
plot(d)
```

```
plot(d, type = "n")
```

```
polygon(d, col = "wheat")
```

```
## Missing values:
```

```
x <- xx <- faithful$eruptions
```

```
x[i.out <- sample(length(x), 10)] <- NA
```

```
doR <- density(x, bw = 0.15, na.rm = TRUE)
```

```
lines(doR, col = "blue")
```

```
points(xx[i.out], rep(0.01, 10))
```

```
## Weighted observations:
```

```
fe <- sort(faithful$eruptions) # has quite a few non-unique values
```

```
## use 'counts / n' as weights:
```

```
dw <- density(unique(fe), weights = table(fe)/length(fe), bw = d$bw)
```

```
utils::str(dw) ## smaller n: only 126, but identical estimate:
```

```
stopifnot(all.equal(d[1:3], dw[1:3]))
```

```
## simulation from a density() fit:
```

```

# a kernel density fit is an equally-weighted mixture.
fit <- density(xx)
N <- 1e6
x.new <- rnorm(N, sample(xx, size = N, replace = TRUE), fit$bw)
plot(fit)
lines(density(x.new), col = "blue")

## The available kernels:
(kernels <- eval(formals(density.default)$kernel))

## show the kernels in the R parametrization
plot(density(0, bw = 1), xlab = "",
     main = "R's density() kernels with bw = 1")
for(i in 2:length(kernels))
  lines(density(0, bw = 1, kernel = kernels[i]), col = i)
legend(1.5, .4, legend = kernels, col = seq(kernels),
      lty = 1, cex = .8, y.intersp = 1)

## show the kernels in the S parametrization
plot(density(0, from = -1.2, to = 1.2, width = 2, kernel = "gaussian"),
     type = "l", ylim = c(0, 1), xlab = "",
     main = "R's density() kernels with width = 1")
for(i in 2:length(kernels))
  lines(density(0, width = 2, kernel = kernels[i]), col = i)
legend(0.6, 1.0, legend = kernels, col = seq(kernels), lty = 1)

##----- Semi-advanced theoretic from here on -----

## Explore the old.coords TRUE --> FALSE change:
set.seed(7); x <- runif(2^12) # N = 4096
den <- density(x) # -> grid of n = 512 points
den0 <- density(x, old.coords = TRUE)
summary(den0$y / den$y) # 1.001 ... 1.011
summary(den0$y / den$y - 1) # ~ 1/(2n-2)
summary(1/ (den0$y / den$y - 1)) # ~ 2n-2 = 1022
corr0 <- 1 - 1/(2*512-2) # 1 - 1/(2n-2)
all.equal(den$y, den0$y * corr0) # ~ 0.0001
plot(den$x, (den0$y - den$y)/den$y, type='o', cex=1/4)
title("relative error of density(runif(2^12), old.coords=TRUE)")
abline(h = 1/1022, v = range(x), lty=2); axis(2, at=1/1022, "1/(2n-2)", las=1)

## The R[K] for our kernels:
(RKs <- cbind(sapply(kernels,
                    function(k) density(kernel = k, give.Rkern = TRUE))))
100*round(RKs["epanechnikov",]/RKs, 4) ## Efficiencies

bw <- bw.SJ(precip) ## sensible automatic choice
plot(density(precip, bw = bw),
     main = "same sd bandwidths, 7 different kernels")
for(i in 2:length(kernels))
  lines(density(precip, bw = bw, kernel = kernels[i]), col = i)

## Bandwidth Adjustment for "Exactly Equivalent Kernels"
h.f <- sapply(kernels, function(k) density(kernel = k, give.Rkern = TRUE))
(h.f <- (h.f["gaussian"] / h.f)^ .2)

```

```
## -> 1, 1.01, .995, 1.007,... close to 1 => adjustment barely visible..

plot(density(precip, bw = bw),
     main = "equivalent bandwidths, 7 different kernels")
for(i in 2:length(kernels))
  lines(density(precip, bw = bw, adjust = h.f[i], kernel = kernels[i]),
        col = i)
legend(55, 0.035, legend = kernels, col = seq(kernels), lty = 1)
```

deriv

```
D (expr, name)
deriv(expr, ...)
deriv3(expr, ...)
```

```
## Default S3 method:
deriv(expr, namevec, function.arg = NULL, tag = ".expr",
      hessian = FALSE, ...)
## S3 method for class 'formula'
deriv(expr, namevec, function.arg = NULL, tag = ".expr",
      hessian = FALSE, ...)
```

```
## Default S3 method:
deriv3(expr, namevec, function.arg = NULL, tag = ".expr",
       hessian = TRUE, ...)
## S3 method for class 'formula'
deriv3(expr, namevec, function.arg = NULL, tag = ".expr",
       hessian = TRUE, ...)
```

```
expr          expressioncallD
namevec       D()
function.arg  NULLTRUEnamevec
tag
hessian
...
```

```
D
derivformulacallexprfunction.argfx
deriv.formuladeriv.default~
deriv3derivhessianTRUEderiv3
+*/*^explogsinacostansinhcoshsqrtprnormdnormasinacosatangammalgamma
trigammapiGamma
log1pexpm1log2log10cospisinpitpnpiGammafactoriallfactorial
```

D

```
derivderiv3expression"gradient"hessianTRUE"hessian"  
function.argNULLderivderiv3
```

nlmoptim

```
## formula argument :  
dx2x <- deriv(~ x^2, "x") ; dx2x  
## Not run: expression({  
  .value <- x^2  
  .grad <- array(0, c(length(.value), 1), list(NULL, c("x")))  
  .grad[, "x"] <- 2 * x  
  attr(.value, "gradient") <- .grad  
  .value  
})  
## End(Not run)  
mode(dx2x)  
x <- -1:2  
eval(dx2x)  
  
## Something 'tougher':  
trig.exp <- expression(sin(cos(x + y^2)))  
( D.sc <- D(trig.exp, "x") )  
all.equal(D(trig.exp[[1]], "x"), D.sc)  
  
( dxy <- deriv(trig.exp, c("x", "y")) )  
y <- 1  
eval(dxy)  
eval(D.sc)  
  
## function returned:  
deriv((y ~ sin(cos(x) * y)), c("x","y"), function.arg = TRUE)  
  
## function with defaulted arguments:  
(fx <- deriv(y ~ b0 + b1 * 2^(-x/th), c("b0", "b1", "th"),  
  function(b0, b1, th, x = 1:7){} ) )  
fx(2, 3, 4)  
  
## First derivative  
  
D(expression(x^2), "x")  
stopifnot(D(as.name("x"), "x") == 1)  
  
## Higher derivatives  
deriv3(y ~ b0 + b1 * 2^(-x/th), c("b0", "b1", "th"),  
  c("b0", "b1", "th", "x") )
```



```

## Higher derivatives:
DD <- function(expr, name, order = 1) {
  if(order < 1) stop("'order' must be >= 1")
  if(order == 1) D(expr, name)
  else DD(D(expr, name), name, order - 1)
}
DD(expression(sin(x^2)), "x", 3)
## showing the limits of the internal "simplify()" :
## Not run:
-sin(x^2) * (2 * x) * 2 + ((cos(x^2) * (2 * x) * (2 * x) + sin(x^2) *
  2) * (2 * x) + sin(x^2) * (2 * x) * 2)

## End(Not run)

## New (R 3.4.0, 2017):
D(quote(log1p(x^2)), "x") ## log1p(x) = log(1 + x)
stopifnot(identical(
  D(quote(log1p(x^2)), "x"),
  D(quote(log(1+x^2)), "x")))
D(quote(expm1(x^2)), "x") ## expm1(x) = exp(x) - 1
stopifnot(identical(
  D(quote(expm1(x^2)), "x") -> Dex1,
  D(quote(exp(x^2)-1), "x"),
  identical(Dex1, quote(exp(x^2) * (2 * x)))))

D(quote(sinpi(x^2)), "x") ## sinpi(x) = sin(pi*x)
D(quote(cospi(x^2)), "x") ## cospi(x) = cos(pi*x)
D(quote(tanpi(x^2)), "x") ## tanpi(x) = tan(pi*x)

stopifnot(identical(D(quote(log2 (x^2)), "x"),
  quote(2 * x/(x^2 * log(2)))),
  identical(D(quote(log10(x^2)), "x"),
  quote(2 * x/(x^2 * log(10)))))

```

deviance

deviance(object, ...)

object

...

object

`df.residualextractAICglm`

`df.residual`

`df.residual(object, ...)`

`object`
`...`

`df.residual`

`x`

`devianceglm`

`diffinv`

`diff`

`diffinv(x, ...)`

```
## Default S3 method:  
diffinv(x, lag = 1, differences = 1, xi, ...)  
## S3 method for class 'ts'  
diffinv(x, lag = 1, differences = 1, xi, ...)
```

`x`
`lag`
`differences`
`xi`
`...`

```
diffinv"ts"default
```

```
"ts"x
```

```
diff
```

```
s <- 1:10  
d <- diff(s)  
diffinv(d, xi = 1)
```

```
dist
```

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

```
as.dist(m, diag = FALSE, upper = FALSE)
```

```
## Default S3 method:
```

```
as.dist(m, diag = FALSE, upper = FALSE)
```

```
## S3 method for class 'dist'
```

```
print(x, diag = NULL, upper = NULL,  
      digits = getOption("digits"), justify = "none",  
      right = TRUE, ...)
```

```
## S3 method for class 'dist'
```

```
as.matrix(x, ...)
```

```
x          "dist"
```

```
method     "euclidean""maximum""manhattan""canberra""binary""minkowski"
```

```
diag       print.dist
```

```
upper      print.dist
```

```
p
```

```
m          "dist""dist"as.matrix()
```

```
digitsjustify formatprint()
```

```
right...
```

xy

euclidean $L_2 \sqrt{\sum_i (x_i - y_i)^2}$

maximum xy

manhattan L_1

canberra $\sum_i |x_i - y_i| / (|x_i| + |y_i|)$
 $x_i + y_i |x_i + y_i| |x_i| + |y_i|$

binary 0NaN

minkowski ppp

InfNaNNA

"dist"as.matrix()as.dist()"dist"

as.dist()"dist"as.matrix()as.matrix()as.dist

dist"dist"

donn <- attr(do, "Size") $i < j \leq ndo[n*(i-1) - i*(i-1)/2 + j-i]n*(n-1)/2n^2$

"class""dist"

Size

Labels

DiagUpper diagupper

call call

method dist()match.arg()method

daisyhclust

require(graphics)

x <- matrix(rnorm(100), nrow = 5)

dist(x)

dist(x, diag = TRUE)

dist(x, upper = TRUE)

m <- as.matrix(dist(x))

d <- as.dist(m)

stopifnot(d == dist(x))

Use correlations between variables "as distance"

dd <- as.dist((1 - cor(USJudgeRatings))/2)

round(1000 * dd) # (prints more nicely)

```

plot(hclust(dd)) # to see a dendrogram of clustered variables

## example of binary and canberra distances.
x <- c(0, 0, 1, 1, 1, 1)
y <- c(1, 0, 1, 1, 0, 1)
dist(rbind(x, y), method = "binary")
## answer 0.4 = 2/5
dist(rbind(x, y), method = "canberra")
## answer 2 * (6/5)

## To find the names
labels(eurodist)

## Examples involving "Inf" :
## 1)
x[6] <- Inf
(m2 <- rbind(x, y))
dist(m2, method = "binary") # warning, answer 0.5 = 2/4
## These all give "Inf":
stopifnot(Inf == dist(m2, method = "euclidean"),
           Inf == dist(m2, method = "maximum"),
           Inf == dist(m2, method = "manhattan"))
## "Inf" is same as very large number:
x1 <- x; x1[6] <- 1e100
stopifnot(dist(cbind(x, y), method = "canberra") ==
           print(dist(cbind(x1, y), method = "canberra")))

## 2)
y[6] <- Inf #-> 6-th pair is excluded
dist(rbind(x, y), method = "binary" ) # warning; 0.5
dist(rbind(x, y), method = "canberra" ) # 3
dist(rbind(x, y), method = "maximum") # 1
dist(rbind(x, y), method = "manhattan") # 2.4

```

Distributions

dxxpxxxqxxrx

[dbeta](#)

[dbinom](#)

[dcauchy](#)

[dchisq](#)

[dexp](#)

[df](#)

[dgamma](#)

[dgeom](#)

[dhyper](#)

dlnorm
dmultinom
dnbinom
dnorm
dpois
dt
dunif
dweibull
pbirthdaysignrankptukeydwilcoxcor.test

RNG
<https://CRAN.R-project.org/view=Distributions>

`dummy.coef`

```
dummy.coef(object, ...)  
  
## S3 method for class 'lm'  
dummy.coef(object, use.na = FALSE, ...)  
  
## S3 method for class 'aovlist'  
dummy.coef(object, use.na = FALSE, ...)  
  
object  
use.na          use.na  
...  
  
contr.helmertcontr.sumdummy.coefcontr.treatment  
poly(x, 2)aov  
  
aov
```

[aovmodel.tables](#)

```
options(contrasts = c("contr.helmert", "contr.poly"))
## From Venables and Ripley (2002) p.165.
npk.aov <- aov(yield ~ block + N*P*K, npk)
dummy.coef(npk.aov)

npk.aovE <- aov(yield ~ N*P*K + Error(block), npk)
dummy.coef(npk.aovE)
```

[ecdf](#)

```
ecdf(x)

## S3 method for class 'ecdf'
plot(x, ..., ylab="Fn(x)", verticals = FALSE,
      col.01line = "gray70", pch = 19)

## S3 method for class 'ecdf'
print(x, digits= getOption("digits") - 2, ...)

## S3 method for class 'ecdf'
summary(object, ...)
## S3 method for class 'ecdf'
quantile(x, ...)
```

xobject	<code>ecdf</code> "ecdf"
...	plot.stepfun plot
ylab	
verticals	plot.stepfun
col.01line	colors
pch	
digits	print

$F_n i / n i$

$x = (x_1, x_2, \dots, x_n) F_n t$

$$F_n(t) = \#\{x_i \leq t\} / n = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{[x_i \leq t]}.$$

`plot.ecdf`[plotecdf](#)[plot.stepfun](#)

```

ecdf"ecdf""stepfun"knots()
summaryobject"header"
quantile(obj, ...)quantile(x, ...)x

"ecdf"

eval(attr(old_obj, "call"), environment(old_obj))

```

stepfunapproxfun splinefun

```

##-- Simple didactical ecdf example :
x <- rnorm(12)
Fn <- ecdf(x)
Fn      # a *function*
Fn(x)   # returns the percentiles for x
tt <- seq(-2, 2, by = 0.1)
12 * Fn(tt) # Fn is a 'simple' function {with values k/12}
summary(Fn)
##--> see below for graphics
knots(Fn) # the unique data values {12 of them if there were no ties}

y <- round(rnorm(12), 1); y[3] <- y[1]
Fn12 <- ecdf(y)
Fn12
knots(Fn12) # unique values (always less than 12!)
summary(Fn12)
summary.stepfun(Fn12)

## Advanced: What's inside the function closure?
ls(environment(Fn12))
## "f"      "method" "na.rm"  "nobs"    "x"      "y"      "yleft"  "yright"
utils::ls.str(environment(Fn12))
stopifnot(all.equal(quantile(Fn12), quantile(y)))

###----- Plotting -----
require(graphics)

op <- par(mfrow = c(3, 1), mgp = c(1.5, 0.8, 0), mar = .1+c(3,3,2,1))

F10 <- ecdf(rnorm(10))
summary(F10)

plot(F10)
plot(F10, verticals = TRUE, do.points = FALSE)

```



```

plot(Fn12 , lwd = 2) ; mtext("lwd = 2", adj = 1)
xx <- unique(sort(c(seq(-3, 2, length.out = 201), knots(Fn12))))
lines(xx, Fn12(xx), col = "blue")
abline(v = knots(Fn12), lty = 2, col = "gray70")

plot(xx, Fn12(xx), type = "o", cex = .1) #- plot.default {ugly}
plot(Fn12, col.hor = "red", add = TRUE) #- plot method
abline(v = knots(Fn12), lty = 2, col = "gray70")
## luxury plot
plot(Fn12, verticals = TRUE, col.points = "blue",
     col.hor = "red", col.vert = "bisque")

##-- this works too (automatic call to ecdf(.)):
plot.ecdf(rnorm(24))
title("via simple plot.ecdf(x)", adj = 1)

par(op)

```

eff.aovlist

eff.aovlist(aovlist)

aovlist aovError

[model.tables.aovlistse.contrast.aovlist](#)

[aovmodel.tables.aovlistse.contrast.aovlist](#)

```
## An example from Yates (1932),
## a 2^3 design in 2 blocks replicated 4 times

Block <- gl(8, 4)
A <- factor(c(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,
             0,1,0,1,0,1,0,1,0,1,0,1))
B <- factor(c(0,0,1,1,0,0,1,1,0,1,0,1,1,0,1,0,
             0,0,1,1,0,0,1,1,0,0,1,1))
C <- factor(c(0,1,1,0,1,0,0,1,0,0,1,1,0,0,1,1,
             1,0,1,0,0,0,1,1,1,1,0,0))
Yield <- c(101, 373, 398, 291, 312, 106, 265, 450, 106, 306, 324, 449,
          272, 89, 407, 338, 87, 324, 279, 471, 323, 128, 423, 334,
          131, 103, 445, 437, 324, 361, 302, 272)
aovdat <- data.frame(Block, A, B, C, Yield)

old <- getOption("contrasts")
options(contrasts = c("contr.helmert", "contr.poly"))

(fit <- aov(Yield ~ A*B*C + Error(Block), data = aovdat))

eff.aovlist(fit)
options(contrasts = old)
```

effects

"lm" "glm"

effects(object, ...)

S3 method for class 'lm'
effects(object, set.sign = FALSE, ...)

object [lm](#)
set.sign TRUE
...

[lmaov](#)^r

[residuals](#)"coef"

^r

coef

```
y <- c(1:3, 7, 5)
x <- c(1:3, 6:7)
( ee <- effects(lm(y ~ x)) )
c( round(ee - effects(lm(y+10 ~ I(x-3.8)))), 3) )
# just the first is different
```

embed

x

```
embed (x, dimension = 1)
```

x
dimension

```
x[t]x[t-1]x[t-dimension+1]txxxx[t]t
```

x

```
x <- 1:10
embed (x, 3)
```

expand.model.frame

na.actionssubsetxsin(x)

```
expand.model.frame(model, extras,
                    envir = environment(formula(model)),
                    na.expand = FALSE)
```

model
extras
envir
na.expand

na.expand = FALSE
model.frame(model) NA
na.action model\$na.action
na.omit na.fail
na.expand = TRUE

model.frame
predict

```
model <- lm(log(Volume) ~ log(Girth) + log(Height), data = trees)
expand.model.frame(model, ~ Girth) # prints data.frame like

dd <- data.frame(x = 1:5, y = rnorm(5), z = c(1,2,NA,4,5))
model <- glm(y ~ x, data = dd, subset = 1:4, na.action = na.omit)
expand.model.frame(model, "z", na.expand = FALSE) # = default
expand.model.frame(model, "z", na.expand = TRUE)
```

Exponential

rate1/rate

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

xq
p
n length(n) > 1
rate
loglog.p
lower.tail $P[X \leq x]P[X > x]$

rate1

λ

$$f(x) = \lambda e^{-\lambda x}$$

$x \geq 0$

dexppepxpqexprexp

nrexp

n

$H(t) = -\log(1 - F(t)) - \text{pexp}(t, r, \text{lower} = \text{FALSE}, \text{log} = \text{TRUE})$

dexppepxpqexp

rexp

[exp](#)

[dgammaadweibull](#)

```
dexp(1) - exp(-1) #-> 0
```

```
## a fast way to generate *sorted* U[0,1] random numbers:
rsunif <- function(n) { n1 <- n+1
  cE <- cumsum(rexp(n1)); cE[seq_len(n)]/cE[n1] }
plot(rsunif(1000), ylim=0:1, pch=".")
abline(0,1/(1000+1), col=adjustcolor(1, 0.5))
```

extractAIC

extractAIC(fit, scale, k = 2, ...)

```

fit          lm
scale        scalestep"lm"scalescale = 0
k            ≡edf
...

```

```

"aov""glm""lm""negbin""coxph""survreg"

```

$$AIC = -2 \log L + k \times ,$$

Ledffit

```

lmaov-2 log LlogLikAICRSSextractAIC-2 log LRSS/s - nC_psn log(RSS/n)AIC
n log(RSS/n) + n + n log 2π - ∑ log wwAICedfextractAIC
glmaic()logLik
k = 2k = log(n)
AIClogLik"lm""glm""negbin"extractAIC

```

```

edf          fit
AIC          fit

```

```

add1drop1step

```

```

AICdevianceadd1step

```

```

utils::example(glm)
extractAIC(glm.D93) #>> 5 15.129

```

factanal

```
factanal(x, factors, data = NULL, covmat = NULL, n.obs = NA,
        subset, na.action, start = NULL,
        scores = c("none", "regression", "Bartlett"),
        rotation = "varimax", control = NULL, ...)
```

```
x          formula
factors
data       model.frameenvironment(formula)
covmat     cov.wt
n.obs      covmat
subset     x
na.action  na.actionxmodel.frame()
start      NULL
scores     "regression""Bartlett"
rotation   functioncharacter"none"loadings
control    list
           nstart start = NULL
           trace FALSE
           lower
           opt optimcontrol
           rotate
...        controlfactanal
```

$$x = \Lambda f + e$$

```
pxp × kΛkfpexΨ
x
```

$$\Sigma = \Lambda \Lambda' + \Psi$$

```
ΛGΛGG
covmatxxdataxcorrelation
startstart = NULLcontrol$nstart - 1
[0,1]control$lower
ffx
```

$$\hat{f} = \Lambda' \Sigma^{-1} x$$

```
fΛ
xNApredict
printloadings
```

```

"factanal"
loadings      "loadings"loadingsprint
uniquenesses
correlation
criteria
factors      factors
dof
method      "mle"
rotmat
scores      napredictna.action
n.obs      NA
call
na.action
STATISTICPVAL

```

```
loadingsprintvarimaxprincompability.covHarman23.corHarman74.cor
```

```

# A little demonstration, v2 is just v1 with noise,
# and same for v4 vs. v3 and v6 vs. v5
# Last four cases are there to add noise
# and introduce a positive manifold (g factor)
v1 <- c(1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,4,5,6)
v2 <- c(1,2,1,1,1,1,2,1,2,1,3,4,3,3,3,4,6,5)
v3 <- c(3,3,3,3,3,1,1,1,1,1,1,1,1,1,1,5,4,6)
v4 <- c(3,3,4,3,3,1,1,2,1,1,1,1,2,1,1,5,6,4)
v5 <- c(1,1,1,1,1,3,3,3,3,3,1,1,1,1,1,6,4,5)
v6 <- c(1,1,1,2,1,3,3,3,4,3,1,1,1,2,1,6,5,4)
m1 <- cbind(v1,v2,v3,v4,v5,v6)
cor(m1)
factanal(m1, factors = 3) # varimax is the default
factanal(m1, factors = 3, rotation = "promax")
# The following shows the g factor as PC1

```



```
prcomp(m1) # signs may depend on platform

## formula interface
factanal(~v1+v2+v3+v4+v5+v6, factors = 3,
         scores = "Bartlett")$scores

## a realistic example from Bartholomew (1987, pp. 61-65)
utils::example(ability.cov)
```

factor.scope

add.scopedrop.scope

add.scope(terms1, terms2)

drop.scope(terms1, terms2)

factor.scope(factor, scope)

terms1

terms2 add.scopedrop.scopedrop.scope

factor "factor"

scope dropadd"factor"

factor.scope

add.scopedrop.scopefactor.scopedropadd

[add1drop1aovlm](#)

```
add.scope( ~ a + b + c + a:b, ~ (a + b + c)^3)
# [1] "a:c" "b:c"
drop.scope( ~ a + b + c + a:b)
# [1] "c" "a:b"
```

family

[glmglm](#)

family(object, ...)

binomial(link = "logit")
gaussian(link = "identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")

link	"link-glm" make.link gaussianidentityloginversebinomiallogitprobitcauchitlogcloglog Gammainverseidentitylogpoissonlogidentitysqrtinverse.gaussian 1/mu^2inverseidentitylog quasilogitprobitcloglogidentityinverselog1/mu^2sqrt power
variance	quasiquasi"constant""mu(1-mu)""mu""mu^2""mu^3"varfunvalidmu dev.residsinitializename
object	familyfamilyglm
...	

family"glm""lm"gaussian()
binomialquasibinomial

\emptyset 1weights

quasibinomialquasipoissonbinomialpoissonglm

"family"

family

link

linkfun

linkinv

variance

dev.resids (y, mu, wt)[residuals](#)

```

aic          NA-2ℓ + 2sℓsglm.fit()AIC()glmlogLik
mu.eta        $\mu = g^{-1}(\eta)\eta d(g^{-1})/d\eta = d\mu/d\eta$ 
initialize   nmustartglm
validmu      TRUEmuvariance
valideta     TRUEetalinkinv
simulate     simulate(object, nsim)"lm"simulatensimnsim
dispersion   NA_real_

```

```

linkvariancelink"link-glm"make.link
link = logitlogitlogit[1]

```

```

quasibinomialquasipoisson

```

```

glmpowermake.link
chooseBinomialNegBinomial

```

```

require(utils) # for str

nf <- gaussian() # Normal family
nf
str(nf)

gf <- Gamma()
gf
str(gf)
gf$linkinv
gf$variance(-3:4) #- == (.)^2

## Binomial with default 'logit' link: Check some properties visually:
bi <- binomial()
et <- seq(-10,10, by=1/8)
plot(et, bi$mu.eta(et), type="l")
## show that mu.eta() is derivative of linkinv() :
lines((et[-1]+et[-length(et)])/2, col=adjustcolor("red", 1/4),
      diff(bi$linkinv(et))/diff(et), type="l", lwd=4)
## which here is the logistic density:
lines(et, dlogis(et), lwd=3, col=adjustcolor("blue", 1/4))
stopifnot(exprs = {
  all.equal(bi$mu.eta(et), dlogis(et))
  all.equal(bi$linkinv(et), plogis(et) -> m)
})

```

```

    all.equal(bi$linkfun(m), qlogis(m))    # logit(.) == qlogis(.) !
  })

## Data from example(glm) :
d.AD <- data.frame(treatment = gl(3,3),
                   outcome    = gl(3,1,9),
                   counts     = c(18,17,15, 20,10,20, 25,13,12))
glm.D93 <- glm(counts ~ outcome + treatment, d.AD, family = poisson())
## Quasipoisson: compare with above / example(glm) :
glm.qD93 <- glm(counts ~ outcome + treatment, d.AD, family = quasipoisson())

glm.qD93
anova (glm.qD93, test = "F")
summary(glm.qD93)
## for Poisson results (same as from 'glm.D93' !) use
anova (glm.qD93, dispersion = 1, test = "Chisq")
summary(glm.qD93, dispersion = 1)


## Example of user-specified link, a logit model for p^days
## See Shaffer, T. 2004. Auk 121(2): 526-540.
logexp <- function(days = 1)
{
  linkfun <- function(mu) qlogis(mu^(1/days))
  linkinv <- function(eta) plogis(eta)^days
  mu.eta <- function(eta) days * plogis(eta)^(days-1) *
    binomial()$mu.eta(eta)
  valideta <- function(eta) TRUE
  link <- paste0("logexp(", days, ")")
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta, name = link),
            class = "link-glm")
}
(bil3 <- binomial(logexp(3)))

## in practice this would be used with a vector of 'days', in
## which case use an offset of 0 in the corresponding formula
## to get the null deviance right.

## Binomial with identity link: often not a good idea, as both
## computationally and conceptually difficult:
binomial(link = "identity") ## is exactly the same as
binomial(link = make.link("identity"))


## tests of quasi
x <- rnorm(100)
y <- rpois(100, exp(1+x))
glm(y ~ x, family = quasi(variance = "mu", link = "log"))
# which is the same as
glm(y ~ x, family = poisson)
glm(y ~ x, family = quasi(variance = "mu^2", link = "log"))
## Not run: glm(y ~ x, family = quasi(variance = "mu^3", link = "log")) # fails
y <- rbinom(100, 1, plogis(x))
# need to set a starting value for the next fit

```

```
glm(y ~ x, family = quasi(variance = "mu(1-mu)", link = "logit"), start = c(0,1))
```

FDist

df1df2ncp

```
df(x, df1, df2, ncp, log = FALSE)
pf(q, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
qf(p, df1, df2, ncp, lower.tail = TRUE, log.p = FALSE)
rf(n, df1, df2, ncp)
```

xq

p

n length(n) > 1

df1df2 Inf

ncp

loglog.p

lower.tail $P[X \leq x]P[X > x]$

df1 = ν_1 df2 = ν_2

$$f(x) = \frac{\Gamma(\nu_1/2 + \nu_2/2)}{\Gamma(\nu_1/2)\Gamma(\nu_2/2)} \left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/2} x^{\nu_1/2-1} \left(1 + \frac{\nu_1 x}{\nu_2}\right)^{-(\nu_1+\nu_2)/2}$$

$x > 0$

$F_{\nu_1,\nu_2}F_{\nu_1,\nu_2}(qF) = 1 - I_x(\nu_2/2,\nu_1/2) = I_{1-x}(\nu_1/2,\nu_2/2), x := \frac{\nu_2}{\nu_2+\nu_1*qF}I_x(a,b)=\text{pbeta}(x,$
 $a,b)$

$\nu_1\nu_2mt_mt_m m$

ncp

dfpfqfrf

NaN

nrf

n

ncp = 0ncpncp

ncpncp

dfdbinomdbeta
pfpbetadf2pchisq
qfqchisqdf2qbeta

dchisqdt

```
## Equivalence of pt(.,nu) with pf(.^2, 1,nu):
x <- seq(0.001, 5, length.out = 100)
nu <- 4
stopifnot(all.equal(2*pt(x,nu) - 1, pf(x^2, 1,nu)),
  ## upper tails:
  all.equal(2*pt(x,      nu, lower.tail=FALSE),
    pf(x^2, 1,nu, lower.tail=FALSE)))

## the density of the square of a t_m is 2*dt(x, m)/(2*x)
# check this is the same as the density of F_{1,m}
all.equal(df(x^2, 1, 5), dt(x, 5)/x)

## Identity (F <-> t): qf(2*p - 1, 1, df) == qt(p, df)^2 for p >= 1/2
p <- seq(1/2, .99, length.out = 50); df <- 10
rel.err <- function(x, y) ifelse(x == y, 0, abs(x-y)/mean(abs(c(x,y))))
stopifnot(all.equal(qf(2*p - 1, df1 = 1, df2 = df),
  qt(p, df)^2))

## Identity (F <-> Beta <-> incompl.beta):
n1 <- 7 ; n2 <- 12; qF <- c((0:4)/4, 1.5, 2:16)
x <- n2/(n2 + n1*qF)
stopifnot(all.equal(pf(qF, n1, n2, lower.tail=FALSE),
  pbeta(x, n2/2, n1/2)))
```

fft

fft(z, inverse = FALSE)
mvfft(z, inverse = FALSE)

z
inverse TRUE+e1/length(x)

```
zfftzy <- fft(z)
```

$$y[h] = \sum_{k=1}^n z[k] \exp(-2\pi i(k-1)(h-1)/n)$$

```
h = 1,...,nlength(y)inverseTRUEexp(-2π...)exp(2π...)
```

```
zfftinverseTRUEy <- fft(z)zfft(y, inverse = TRUE) / length(y)
```

```
mvfft
```

[convolvenextn](#)

```
x <- 1:4
```

```
fft(x)
```

```
fft(fft(x), inverse = TRUE)/length(x)
```

```
## Slow Discrete Fourier Transform (DFT) - e.g., for checking the formula
```

```
fft0 <- function(z, inverse=FALSE) {
```

```
  n <- length(z)
```

```
  if(n == 0) return(z)
```

```
  k <- 0:(n-1)
```

```
  ff <- (if(inverse) 1 else -1) * 2*pi * 1i * k/n
```

```
  vapply(1:n, function(h) sum(z * exp(ff*(h-1))), complex(1))
```

```
}
```

```
reld <- function(x,y) 2* abs(x - y) / abs(x + y)
```

```
n <- 2^8
```

```
z <- complex(n, rnorm(n), rnorm(n))
```

```
## relative differences in the order of 4*10^{-14} :
```

```
summary(reld(fft(z), fft0(z)))
```

```
summary(reld(fft(z, inverse=TRUE), fft0(z, inverse=TRUE)))
```

`filter`

```
filter(x, filter, method = c("convolution", "recursive"),
      sides = 2, circular = FALSE, init)
```

```

x
filter
method      "convolution""recursive""convolution""recursive"
sides        sides = 1sides = 2
circular     TRUENA
init

```

```

xfilter

```

$$y_i = x_i + f_1 y_{i-1} + \cdots + f_p y_{i-p}$$

$$y_i = f_1 x_{i+o} + \cdots + f_p x_{i+o-(p-1)}$$

```

osides

```

```

convolve(, type = "filter")filter

```

```

convolvearima.sim

```

```

x <- 1:100
filter(x, rep(1, 3))
filter(x, rep(1, 3), sides = 1)
filter(x, rep(1, 3), sides = 1, circular = TRUE)

filter(presidents, rep(1, 3))

```

```

fisher.test

```

```

fisher.test(x, y = NULL, workspace = 200000, hybrid = FALSE,
            hybridPars = c(expect = 5, percent = 80, Emin = 1),
            control = list(), or = 1, alternative = "two.sided",
            conf.int = TRUE, conf.level = 0.95,
            simulate.p.value = FALSE, B = 2000)

```



```

x
y          x
workspace  2 × 2simulate.p.values = TRUE
hybrid     2 × 2
hybridPars
control    "mult" ≥ 22 × 2fexact.c
or         2 × 2
alternative "two.sided""greater""less"2 × 2
conf.int   2 × 2
conf.level 2 × 2conf.int = TRUE
simulate.p.value
           2 × 2
B          simulate.p.value

```

```

xxy
2 × 2FEXACThttps://netlib.org/toms/643231 − 1
2 × 2alternative = "greater"or
2 × 2hybrid = TRUEhybridPars = c(expect = 5, percent = 80, Emin = 1)1= Emin=
percent= expectif()
r × cr > 2c > 2workspacesimulate.p.value = TRUE
B = 20001/(B + 1)

```

```

"htest"
p.value
conf.int  2 × 2conf.int = TRUE
estimate  2 × 2
null.value or2 × 2
alternative
method     "Fisher's Exact Test for Count Data"
data.name

```

```

r × c
r × c
r × c

```

`chisq.test`

`fisher.exact2 × 2`

```
## Agresti (1990, p. 61f; 2002, p. 91) Fisher's Tea Drinker
## A British woman claimed to be able to distinguish whether milk or
## tea was added to the cup first. To test, she was given 8 cups of
## tea, in four of which milk was added first. The null hypothesis
## is that there is no association between the true order of pouring
## and the woman's guess, the alternative that there is a positive
## association (that the odds ratio is greater than 1).
TeaTasting <-
matrix(c(3, 1, 1, 3),
      nrow = 2,
      dimnames = list(Guess = c("Milk", "Tea"),
                      Truth = c("Milk", "Tea")))
fisher.test(TeaTasting, alternative = "greater")
## => p = 0.2429, association could not be established

## Fisher (1962, 1970), Criminal convictions of like-sex twins
Convictions <- matrix(c(2, 10, 15, 3), nrow = 2,
                    dimnames =
                      list(c("Dizygotic", "Monozygotic"),
                          c("Convicted", "Not convicted")))
Convictions
fisher.test(Convictions, alternative = "less")
fisher.test(Convictions, conf.int = FALSE)
fisher.test(Convictions, conf.level = 0.95)$conf.int
fisher.test(Convictions, conf.level = 0.99)$conf.int

## A r x c table Agresti (2002, p. 57) Job Satisfaction
Job <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4,
             dimnames = list(income = c("< 15k", "15-25k", "25-40k", "> 40k"),
                             satisfaction = c("VeryD", "LittleD", "ModerateS", "VeryS")))
fisher.test(Job) # 0.7827
fisher.test(Job, simulate.p.value = TRUE, B = 1e5) # also close to 0.78

## 6th example in Mehta & Patel's JASA paper
MP6 <- rbind(
  c(1,2,2,1,1,0,1),
  c(2,0,0,2,3,0,0),
  c(0,1,1,1,2,7,3),
  c(1,1,2,0,0,0,1),
  c(0,1,1,1,1,0,0))
fisher.test(MP6)
# Exactly the same p-value, as Cochran's conditions are not met:
fisher.test(MP6, hybrid = TRUE)
```

fitted

fittedfitted.values
fittedfittedfitted.values
[napredictnls](#)

fitted(object, ...)
fitted.values(object, ...)

object
...

object

[coefficientsglmmlmresiduals](#)

fivenum

fivenum(x, na.rm = TRUE)

x	NA±Inf
na.rm	TRUE NANaN

[boxplot.stats](#)

[IQRboxplot.statsmedianquantilerange](#)

fivenum(c(rnorm(100), -1:1/0))

fligner.test

```
fligner.test(x, ...)
```

```
## Default S3 method:  
fligner.test(x, g, ...)
```

```
## S3 method for class 'formula'  
fligner.test(formula, data, subset, na.action, ...)
```

```
x  
g                xx  
formula          lhs ~ rhs|lhsrhs  
data             model.frame(formula, data, subset)  
subset  
na.action        NAgetOption("na.action")  
...
```

```
xgfligner.test(x)fligner.test(list(x, ...))
```

```
xgxx
```

```
 $ka(i) = qnorm((1 + i/(n + 1))/2)X^2$ 
```

```
"htest"
```

```
statistic         $X^2$ 
```

```
parameter
```

```
p.value
```

```
method           "Fligner-Killeen test of homogeneity of variances"
```

```
data.name
```

[ansari.testmood.testvar.testbartlett.test](#)

```
require(graphics)
```

```
plot(count ~ spray, data = InsectSprays)  
fligner.test(InsectSprays$count, InsectSprays$spray)  
fligner.test(count ~ spray, data = InsectSprays)  
## Compare this to bartlett.test()
```

formula

```
formula
as.formulaobject"formula"

formula(x, ...)
DF2formula(x, env = parent.frame())
as.formula(object, env = parent.frame())

## S3 method for class 'formula'
print(x, showEnv = !identical(e, .GlobalEnv), ...)

xobject      DF2formula()data.frame
...
env
showEnv

lmglm~y ~ modelymodel+:
+:
  *a*ba + b + a:b
  ^ (a+b+c)^2(a+b+c)*(a+b+c)abc
  %in%a + b %in% aa + a:b
  /a / ba + b %in% a
  -(a+b+c)^2 - a:ba + b + c + b:c + a:cy ~ x - 1y ~ x + 0y ~ 0 + x
log(y) ~ a + log(x)
I()y ~ a + I(b+c)b+cbc
`like this`
offsetstrataclusterspecialsterms.formula
.dataterms.formulaupdate.formula
formula"nls""formula""terms"formula"formula"
formula"terms"model.frame()≤DF2formula()"terms"+

"formula"

model.framedata
~as.formulaenv
```

```

characterxx
eval(call("~", quote(foo + bar)))formula(c("~", "foo + bar"))
eval()

```

```

~Ioffset
update.formulaterms.formulaall.varslmglmcplotreformulate

```

```

class(fo <- y ~ x1*x2) # "formula"
fo
typeof(fo) # R internal : "language"
terms(fo)

environment(fo)
environment(as.formula("y ~ x"))
environment(as.formula("y ~ x", env = new.env()))

## Create a formula for a model with a large number of variables:
xnam <- paste0("x", 1:25)
(fmla <- as.formula(paste("y ~ ", paste(xnam, collapse= "+"))))
## Equivalent with reformulate():
fmla2 <- reformulate(xnam, response = "y")
stopifnot(identical(fmla, fmla2))

```

formula.nls	nls
-------------	-----

object

```

## S3 method for class 'nls'
formula(x, ...)

```

```

x          "nls"
...

```

object

nlsformula

```
fm1 <- nls(circumference ~ A/(1+exp((B-age)/C)), Orange,  
          start = list(A = 160, B = 700, C = 350))  
formula(fm1)
```

friedman.test

```
friedman.test(y, ...)
```

```
## Default S3 method:
```

```
friedman.test(y, groups, blocks, ...)
```

```
## S3 method for class 'formula'
```

```
friedman.test(formula, data, subset, na.action, ...)
```

```
y
```

```
groups      yy
```

```
blocks      yy
```

```
formula     a ~ b | cabc
```

```
data        model.frameformulaenvironment(formula)
```

```
subset
```

```
na.action   NAgetOption("na.action")
```

```
...
```

```
friedman.testygroupsblocks
```

```
blocksygroups
```

```
ygroupsblocksNAgroupsblocksyNA
```

```
"htest"
```

```
statistic
```

```
parameter
```

```
p.value
```

```
method      "Friedman rank sum test"
```

```
data.name
```

quade.test

```
## Hollander & Wolfe (1973), p. 140ff.
## Comparison of three methods ("round out", "narrow angle", and
## "wide angle") for rounding first base. For each of 18 players
## and the three method, the average time of two runs from a point on
## the first base line 35ft from home plate to a point 15ft short of
## second base is recorded.
RoundingTimes <-
matrix(c(5.40, 5.50, 5.55,
        5.85, 5.70, 5.75,
        5.20, 5.60, 5.50,
        5.55, 5.50, 5.40,
        5.90, 5.85, 5.70,
        5.45, 5.55, 5.60,
        5.40, 5.40, 5.35,
        5.45, 5.50, 5.35,
        5.25, 5.15, 5.00,
        5.85, 5.80, 5.70,
        5.25, 5.20, 5.10,
        5.65, 5.55, 5.45,
        5.60, 5.35, 5.45,
        5.05, 5.00, 4.95,
        5.50, 5.50, 5.40,
        5.45, 5.55, 5.50,
        5.55, 5.55, 5.35,
        5.45, 5.50, 5.55,
        5.50, 5.45, 5.25,
        5.65, 5.60, 5.40,
        5.70, 5.65, 5.55,
        6.30, 6.30, 6.25),
      nrow = 22,
      byrow = TRUE,
      dimnames = list(1 : 22,
                      c("Round Out", "Narrow Angle", "Wide Angle")))
friedman.test(RoundingTimes)
## => strong evidence against the null that the methods are equivalent
## with respect to speed

wb <- aggregate(warpbreaks$breaks,
                by = list(w = warpbreaks$wool,
                          t = warpbreaks$tension),
                FUN = mean)

wb
friedman.test(wb$x, wb$w, wb$t)
friedman.test(x ~ w | t, data = wb)
```

ftable

```
ftable(x, ...)
```

```
## Default S3 method:
```

```
ftable(..., exclude = c(NA, NaN), row.vars = NULL,  
        col.vars = NULL)
```

```
x...          "table""ftable"
```

```
exclude       factor
```

```
row.vars
```

```
col.vars
```

```
ftablerow.varscol.varsprint.ftable"ftable"
```

```
ftableleftable.defaultrow.varscol.vars"table""ftable"as.tableftablerow.vars
```

```
col.vars
```

```
table
```

```
ftable.formula
```

```
as.tableas.matrixas.data.frame
```

```
ftable"ftable""row.vars""col.vars"
```

```
ftable.formuladata = .read.ftabletablextabs
```

```
## Start with a contingency table.
```

```
ftable(Titanic, row.vars = 1:3)
```

```
ftable(Titanic, row.vars = 1:2, col.vars = "Survived")
```

```
ftable(Titanic, row.vars = 2:1, col.vars = "Survived")
```

```
## Start with a data frame.
```

```
x <- ftable(mtcars[c("cyl", "vs", "am", "gear")])
```

```
x
```

```
ftable(x, row.vars = c(2, 4))
```

```
## Start with expressions, use table()'s "dnn" to change labels
```

```
ftable(mtcars$cyl, mtcars$vs, mtcars$am, mtcars$gear, row.vars = c(2, 4),  
        dnn = c("Cylinders", "V/S", "Transmission", "Gears"))
```

`fable.formula`

```
## S3 method for class 'formula'
fable(formula, data = NULL, subset, na.action, ...)
```

```
formula
data      model.frame
subset    data
na.action NAdata
...       fablefable.default
```

```
fable
formula+.
data"table""fable"na.actionsubset
formula
```

```
fablefable.defaulttable
```

```
Titanic
x <- fable(Survived ~ ., data = Titanic)
x
fable(Sex ~ Class + Age, data = x)
```

`GammaDist`

`shapyscale`

```
dgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pgamma(q, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
qgamma(p, shape, rate = 1, scale = 1/rate, lower.tail = TRUE,
       log.p = FALSE)
rgamma(n, shape, rate = 1, scale = 1/rate)
```

`xq`
`p`
`n` `length(n) > 1`
`rate`
`shapescal` `scale`
`loglog.p` `TRUE` $p\log(p)$
`lower.tail` $P[X \leq x]P[X > x]$

`scale1`
`shape=` α `scale=` σ

$$f(x) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\sigma}$$

$x \geq 0$ $\alpha > 0$ $\sigma > 0$ $\Gamma(\alpha)$ `gamma()` $a = 0$

$E(X) = \alpha\sigma$ $Var(X) = \alpha\sigma^2$

$H(t) = -\log(1 - F(t))$

`-pgamma(t, ..., lower = FALSE, log = TRUE)`

`shapescal` x `rgammascalescale = 1`

`dgamma``pgamma``qgamma``mgamma`

`NaN`

`nrgamma`

`n`

`shaperatescalescalerate`

`pgamma`

$$P(a, x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt$$

$P(a, x)$ `pgamma(x, a)` $\gamma(a, x)$ $\gamma(a, x) = \int_0^x t^{a-1} e^{-t} dt$ `pgamma(x, a) * gamma(a)`

$$\Gamma(a, x) = \int_x^\infty t^{a-1} e^{-t} dt,$$

`pgamma(x, a, lower = FALSE) * gamma(a)`

`pgamma(x, a, ...)` $a > 0$ `gamma_inc(a, x)` $\Gamma(a, x)$

https://en.wikipedia.org/wiki/Incomplete_gamma_function<https://dlmf.nist.gov/8.2#i>

[dgamma](#)[dbinom](#)

[pgamma](#)

[qgamma](#)

`rgamma``shape >= 1`

`0 < shape < 1`

<https://dlmf.nist.gov/>

[gamma](#)

[dbetadchisq](#)

```
-log(dgamma(1:4, shape = 1))
p <- (1:9)/10
pgamma(qgamma(p, shape = 2), shape = 2)
1 - 1/exp(qgamma(p, shape = 1))

# even for shape = 0.001 about half the mass is on numbers
# that cannot be represented accurately (and most of those as zero)
pgamma(.Machine$double.xmin, 0.001)
pgamma(5e-324, 0.001) # on most machines 5e-324 is the smallest
                        # representable non-zero number
table(rgamma(1e4, 0.001) == 0)/1e4
```

Geometric

`prob`

```
dgeom(x, prob, log = FALSE)
pgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
qgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
rgeom(n, prob)
```

xq
p
n length(n) > 1
prob 0 < prob <= 1
loglog.p
lower.tail $P[X \leq x]P[X > x]$

prob= p
$$p(x) = p(1 - p)^x$$

$x = 0, 1, 2, \dots, 0 < p \leq 1$

xdgeom
 $x F(x) \geq p F$

dgeom pgeom qgeom rgeom
probNaN
nrgeom
n
rgeomdouble

dgeomdbinomdbinom
pgeomqgeom
rgeom

dnbinom

qgeom((1:9)/10, prob = .2)
Ni <- rgeom(20, prob = 1/4); table(factor(Ni, 0:max(Ni)))

`getInitial`

`datapframeparametersselfStartinitial``getInitial(object, data, ...)``object selfStart``data selfStart``...``selfStart``nlselfStartselfStart.defaultselfStart.formulanlsList`

```
PurTrt <- Puromycin[ Puromycin$state == "treated", ]  
print(getInitial( rate ~ SSmicmen( conc, Vm, K ), PurTrt ), digits = 3)
```

`glm`

`glm`

```
glm(formula, family = gaussian, data, weights, subset,  
     na.action, start = NULL, etastart, mustart, offset,  
     control = list(...), model = TRUE, method = "glm.fit",  
     x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL, ...)
```

```
glm.fit(x, y, weights = rep.int(1, nobs),  
        start = NULL, etastart = NULL, mustart = NULL,  
        offset = rep.int(0, nobs), family = gaussian(),  
        control = list(), intercept = TRUE, singular.ok = TRUE)
```

```
## S3 method for class 'glm'  
weights(object, type = c("prior", "working"), ...)
```

formula	"formula"
family	glmglm.fitfamily
data	as.data.frameenvironment(formula)glm
weights	NULL
subset	
na.action	NAna.actionoptionsna.failna.omitNULLna.exclude
start	
etastart	
mustart	
offset	NULLoffsetmodel.offset
control	glm.fitglm.control
model	
method	"glm.fit""model.frame" glm.fit
xy	glm glm.fitxn * pyn
singular.ok	FALSE
contrasts	contrasts.argmodel.matrix.default
intercept	
object	"glm"
type	
...	glmcontrol weights

```

response ~ termsresponsebinomialquasibinomialfactorfirst + second
firstsecond
first:secondfirstsecondfirst*secondfirstsecondfirst + second + first:second
terms
NULLweightsweightsweights $w_i y_i w_i$ 
glm.fit
etastartstartmustartquasigaussian("log")
weightssubsetoffsetetastartmustartformuladataformulamean(x)subsetmodel.frame

```

```

glm"glm""lm"method
summarysummary.glmANOVAANOVA.glm
coefficientseffectsvaluesresidualsglm
weightsna.action
"glm"

```

```

coefficients
residuals      NA
fitted.values
rank
family         family
linear.predictors

deviance
aic            aicNA
null.deviance  deviance
iter
weights
prior.weights  1
df.residual
df.null
y             y
x
model
converged
boundary
call
formula
terms         terms
data          data argument
offset
control       control
method        characterglm()function
contrasts
xlevels
na.action     model.frameNA

qrReffects
"glm"c("glm", "lm")"lm""lm""glm"residualsweights
binomialglmprior.weightsy

methodglm.fitglm.fit
glm

glm

```



```

anova.glmsummary.glmglmmanovasummaryeffectsfitted.valuesresiduals
lm
loglinloglm
bigglm
esophinfertpredict.glm

```

```

## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
data.frame(treatment, outcome, counts) # showing data
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
anova(glm.D93)
summary(glm.D93)
## Computing AIC [in many ways]:
(A0 <- AIC(glm.D93))
(l1 <- logLik(glm.D93))
A1 <- -2*c(l1) + 2*attr(l1, "df")
A2 <- glm.D93$family$aic(counts, mu=fitted(glm.D93), wt=1) +
      2 * length(coef(glm.D93))
stopifnot(exprs = {
  all.equal(A0, A1)
  all.equal(A1, A2)
  all.equal(A1, glm.D93$aic)
})

```

```

## an example with offsets from Venables & Ripley (2002, p.189)
utils::data(anorexia, package = "MASS")

```

```

anorex.1 <- glm(Postwt ~ Prewt + Treat + offset(Prewt),
  family = gaussian, data = anorexia)
summary(anorex.1)

```

```

# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
  u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(118,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12))
summary(glm(lot1 ~ log(u), data = clotting, family = Gamma))
summary(glm(lot2 ~ log(u), data = clotting, family = Gamma))
## Aliased ("S"ingular) -> 1 NA coefficient
(fS <- glm(lot2 ~ log(u) + log(u^2), data = clotting, family = Gamma))

```

```

tools::assertError(update(fs, singular.ok=FALSE), verbose=interactive())
## -> .. "singular fit encountered"

## Not run:
## for an example of the use of a terms object as a formula
demo(glm.vr)

## End(Not run)

```

glm.control

[glmglm.fit](#)control

```
glm.control(epsilon = 1e-8, maxit = 25, trace = FALSE)
```

```

epsilon       $\epsilon |dev - dev_{old}| / (|dev| + 0.1) < \epsilon$ 
maxit
trace

```

```

controlglmcontrolglm.fitglm.control
epsilon10-10
tracecatoptions(digits = *)

```

[glm.fit](#)glm

```

### A variation on example(glm) :

## Annette Dobson's example ...
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
oo <- options(digits = 12) # to see more when tracing :
glm.D93X <- glm(counts ~ outcome + treatment, family = poisson(),
               trace = TRUE, epsilon = 1e-14)
options(oo)
coef(glm.D93X) # the last two are closer to 0 than in ?glm's glm.D93

```

glm.summaries

[methods](#)glmsummary.glm

```
## S3 method for class 'glm'
family(object, ...)
```

```
## S3 method for class 'glm'
residuals(object, type = c("deviance", "pearson", "working",
                           "response", "partial"), ...)
```

```
object      glmglm
type        "deviance""pearson""working""response""partial"
...
```

```
residualsna.actionna.action = na.omitna.action = na.excludeNAnaresid
y = FALSE
```

[glm](#)[glm.obj](#)[anova.glm](#)[summary.glm](#)[coef](#)[deviance](#)[cdf.residual](#)[effects](#)[fitted](#)[residuals](#)
[rstandard](#)[rstudent](#)

hclust

```
hclust(d, method = "complete", members = NULL)
```

```
## S3 method for class 'hclust'
plot(x, labels = NULL, hang = 0.1, check = TRUE,
     axes = TRUE, frame.plot = FALSE, ann = TRUE,
     main = "Cluster Dendrogram",
     sub = NULL, xlab = NULL, ylab = "Height", ...)
```

```

d          dist
method     "ward.D""ward.D2""single""complete""average""mcquitty""median"
          "centroid"
members    NULLd
x          hclust
hang
check      xxhclust()check=TRUE
labels     labels = FALSE
axesframe.plotann
          plot.default
mainsubxlabylab
          titlesubxlabtree$call
...        cextext

```

```

n
"median""centroid"
"ward.D""ward"≤"ward.D2"agnes(*, method="ward")hclust(*, "ward.D2")
members != NULLdmembershclust
nn − 12(n−1)hclust

```

```

"hclust"

merge      n − 1mergeij−jjjmerge
height     n − 1method
order      merge
labels
call
method
dist.method d"method"

printplotidentifyidentify.hclustrect.hclust()hclust

```

```

"centroid"dist(*)^2

```

```

hclust

```

[identify.hclustrect.hclustcutreedendrogramkmeans](#)
[agnes](#)

```
require(graphics)

### Example 1: Violent crime rates by US state

hc <- hclust(dist(USArrests), "ave")
plot(hc)
plot(hc, hang = -1)

## Do the same with centroid clustering and *squared* Euclidean distance,
## cut the tree into ten clusters and reconstruct the upper part of the
## tree from the cluster centers.
hc <- hclust(dist(USArrests)^2, "cen")
memb <- cutree(hc, k = 10)
cent <- matrix(numeric(), 10, 4)
for(k in 1:10)
  cent[k,] <- colMeans(USArrests[memb == k, , drop = FALSE])

hc1 <- hclust(dist(cent)^2, method = "cen", members = table(memb))
opar <- par(mfrow = c(1, 2))
plot(hc, labels = FALSE, hang = -1, main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1, main = "Re-start from 10 clusters")
par(opar)

### Example 2: Straight-line distances among 10 US cities
## Compare the results of algorithms "ward.D" and "ward.D2"

mds2 <- -cmdscale(UScitiesD)
plot(mds2, type="n", axes=FALSE, ann=FALSE)
text(mds2, labels=rownames(mds2), xpd = NA)

hcity.D <- hclust(UScitiesD, "ward.D") # "wrong"
hcity.D2 <- hclust(UScitiesD, "ward.D2")
opar <- par(mfrow = c(1, 2))
plot(hcity.D, hang=-1)
plot(hcity.D2, hang=-1)
par(opar)
```

heatmap

[image](#)(t(x))

```
heatmap(x, Rowv = NULL, Colv = if(symm)"Rowv" else NULL,
        distfun = dist, hclustfun = hclust,
        reorderfun = function(d, w) reorder(d, w),
        add.expr, symm = FALSE, revC = identical(Colv, "Rowv"),
        scale = c("row", "column", "none"), na.rm = TRUE,
        margins = c(5, 5), ColSideColors, RowSideColors,
        cexRow = 0.2 + 1/log10(nr), cexCol = 0.2 + 1/log10(nc),
        labRow = NULL, labCol = NULL, main = NULL,
        xlab = NULL, ylab = NULL,
        keep.dendro = FALSE, verbose = getOption("verbose"), ...)
```

x	
Rowv	dendrogram NA NULL
Colv	RowvColv = "Rowv"
distfun	dist
hclustfun	RowvColv hclust distfun as.dendrogram
reorderfun	function(d, w) reorder.dendrogram
add.expr	image
symm	xx
revC	rev
scale	"row"symm"none"
na.rm	NA
margins	par (mar = *)
ColSideColors	ncol(x)x
RowSideColors	nrow(x)x
cexRowcexCol	cex.axis
labRowlabCol	rownames(x)colnames(x)
mainxlabylab	
keep.dendro	RowvColv
verbose	
...	image col

```
RowvColvdd <- as.dendrogram(hclustfun(distfun(X)))Xxt(x)
reorder(dd, Rowv)Rowv <- rowMeans(x, na.rm = na.rm)NA
scale = "row"
```

```

rowInd      order.dendrogram
colInd
Rowv        Rowvkeep.dendro
Colv        Colvkeep.dendro

```

```

Rowv = NAColw = NAorder.dendrogram(Rowv)Rowvreorder()
heatmap(layoutimagepar(mfrow = *)(mfcol = *))

```

imagehclust

```

require(graphics); require(grDevices)
x <- as.matrix(mtcars)
rc <- rainbow(nrow(x), start = 0, end = .3)
cc <- rainbow(ncol(x), start = 0, end = .3)
hv <- heatmap(x, col = cm.colors(256), scale = "column",
              RowSideColors = rc, ColSideColors = cc, margins = c(5,10),
              xlab = "specification variables", ylab = "Car Models",
              main = "heatmap(<Mtcars data>, ..., scale = \"column\")")
utils::str(hv) # the two re-ordering index vectors

## no column dendrogram (nor reordering) at all:
heatmap(x, Colv = NA, col = cm.colors(256), scale = "column",
        RowSideColors = rc, margins = c(5,10),
        xlab = "specification variables", ylab = "Car Models",
        main = "heatmap(<Mtcars data>, ..., scale = \"column\")")

## "no nothing"
heatmap(x, Rowv = NA, Colv = NA, scale = "column",
        main = "heatmap(*, NA, NA) ~ image(t(x))")

## Demonstration of the 'scale' argument:
## The only change in the code is the 'scale' arg.
## The only visible change is in the color scale on the heatmap
## (the original data are not scaled).

heatmap(x, col = terrain.colors(128), scale = "column",
        RowSideColors = rc,
        ColSideColors = cc,
        margins = c(5,10),
        main = "heatmap(<Mtcars data>, ..., scale = \"column\")")
heatmap(x, col = terrain.colors(128), scale = "none",
        RowSideColors = rc,
        ColSideColors = cc,
        margins = c(5,10),

```

```

main = "heatmap(<Mtcars data>, ..., scale = \"none\")")

round(Ca <- cor(attitude), 2)
symnum(Ca) # simple graphic
heatmap(Ca, symm = TRUE, margins = c(6,6)) # with reorder()
heatmap(Ca, Rowv = FALSE, symm = TRUE, margins = c(6,6)) # _NO_ reorder()
## slightly artificial with color bar, without and with ordering:
cc <- rainbow(nrow(Ca))
heatmap(Ca, Rowv = FALSE, symm = TRUE, RowSideColors = cc, ColSideColors = cc,
  margins = c(6,6))
heatmap(Ca, symm = TRUE, RowSideColors = cc, ColSideColors = cc,
  margins = c(6,6))

## For variable clustering, rather use distance based on cor():
symnum( cU <- cor(USJudgeRatings) )

hU <- heatmap(cU, Rowv = FALSE, symm = TRUE, col = topo.colors(16),
  distfun = function(c) as.dist(1 - c), keep.dendro = TRUE)
## The Correlation matrix with same reordering:
round(100 * cU[hU[[1]], hU[[2]])
## The column dendrogram:
utils::str(hU$Colv)

```

HoltWinters

```

HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,
  seasonal = c("additive", "multiplicative"),
  start.periods = 2, l.start = NULL, b.start = NULL,
  s.start = NULL,
  optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),
  optim.control = list())

```

x	ts
alpha	<i>alpha</i>
beta	<i>beta</i> FALSE
gamma	<i>gamma</i> FALSE
seasonal	"additive""multiplicative"gamma
start.periods	
l.start	
b.start	
s.start	$s_1[0] \dots s_p[0]$
optim.start	alphabeta gamma
optim.control	optim

$$\hat{Y}[t+h] = a[t] + hb[t] + s[t-p+1+(h-1) \bmod p],$$

$$a[t]b[t]s[t]$$

$$a[t] = \alpha(Y[t] - s[t-p]) + (1-\alpha)(a[t-1] + b[t-1])$$

$$b[t] = \beta(a[t] - a[t-1]) + (1-\beta)b[t-1]$$

$$s[t] = \gamma(Y[t] - a[t]) + (1-\gamma)s[t-p]$$

$$\hat{Y}[t+h] = (a[t] + hb[t]) \times s[t-p+1+(h-1) \bmod p].$$

$$a[t]b[t]s[t]$$

$$a[t] = \alpha(Y[t]/s[t-p]) + (1-\alpha)(a[t-1] + b[t-1])$$

$$b[t] = \beta(a[t] - a[t-1]) + (1-\beta)b[t-1]$$

$$s[t] = \gamma(Y[t]/a[t]) + (1-\gamma)s[t-p]$$

x
 $\alpha\beta\gamma$ NULLoptimizeoptim
 absdecomposestart.periodsabx[2]x[2] - x[1]ax[1]

"HoltWinters"
 fitted
 x
 alpha
 beta
 gamma
 coefficients a, b, s1, ..., sp
 seasonal seasonal
 SSE
 call

<David.Meyer@wu.ac.at>

predict.HoltWintersoptim

```

require(graphics)

## Seasonal Holt-Winters
(m <- HoltWinters(co2))
plot(m)
plot(fitted(m))

(m <- HoltWinters(AirPassengers, seasonal = "mult"))
plot(m)

## Non-Seasonal Holt-Winters
x <- uspop + rnorm(uspop, sd = 5)
m <- HoltWinters(x, gamma = FALSE)
plot(m)

## Exponential Smoothing
m2 <- HoltWinters(x, gamma = FALSE, beta = FALSE)
lines(fitted(m2)[,1], col = 3)

```

Hypergeometric

```

dhyper(x, m, n, k, log = FALSE)
phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
rhyper(nn, m, n, k)

```

```

xq
m
n
k          0, 1, ..., m + n
p
nn          length(nn) > 1
loglog.p
lower.tail   $P[X \leq x]P[X > x]$ 

```

$$mnkNpN - NpnN := m + n$$

$$p(x) = \binom{m}{x} \binom{n}{k-x} / \binom{m+n}{k}$$

$x = 0, \dots, k$

$$p(x) \max(0, k - n) \leq x \leq \min(k, m)$$

$$p := m/(m+n)Np = N \times p$$

$$E[X] = \mu = kp$$

$$(X) = kp(1-p)\frac{m+n-k}{m+n-1},$$

$$(k,p)k=1$$

$$xF(x) \geq pF$$

`rhyper()`*m,n,k*[Machine\\$integer.max](#)`qhyper(runif(nn), m,n,k)`

`dhyper``phyper``qhyper``rrhyper`

`NaN`

`nrhyper`

`n`

`dhyper`[dbinom](#)

`phyper``dhyper``phyper(...)/dhyper(...)`

`qhyper``phyper()`

`rhyper`

```
m <- 10; n <- 7; k <- 8
x <- 0:(k+1)
rbind(phyper(x, m, n, k), dhyper(x, m, n, k))
all(phyper(x, m, n, k) == cumsum(dhyper(x, m, n, k))) # FALSE
## but errors are very small:
signif(phyper(x, m, n, k) - cumsum(dhyper(x, m, n, k)), digits = 3)

stopifnot(abs(phyper(x, m, n, k) - cumsum(dhyper(x, m, n, k))) < 5e-16)
```

identify.hclust

identify.hclust

```
## S3 method for class 'hclust'
identify(x, FUN = NULL, N = 20, MAXCLUSTER = 20, DEV.FUN = NULL,
        ...)
```

```
x          hclust
FUN
N
MAXCLUSTER
DEV.FUN     FUN
...         FUN
```

[invisible](#)

FUNNULLFUNDEV.FUN

[identify](#)

FUN

[hclustrect.hclust](#)

```
## Not run:
require(graphics)

hca <- hclust(dist(USArrests))
plot(hca)
(x <- identify(hca)) ## Terminate with 2nd mouse button !!

hci <- hclust(dist(iris[,1:4]))
plot(hci)
identify(hci, function(k) print(table(iris[k,5])))

# open a new device (one for dendrogram, one for bars):
dev.new() # << make that narrow (& small)
          # and *beside* 1st one
nD <- dev.cur()          # to be for the barplot
dev.set(dev.prev())      # old one for dendrogram
plot(hci)
## select subtrees in dendrogram and "see" the species distribution:
identify(hci, function(k) barplot(table(iris[k,5]), col = 2:4), DEV.FUN = nD)

## End(Not run)
```

influence.measures

```
influence.measures(model, infl = influence(model))

rstandard(model, ...)
## S3 method for class 'lm'
rstandard(model, infl = lm.influence(model, do.coef = FALSE),
           sd = sqrt(deviance(model)/df.residual(model)),
           type = c("sd.1", "predictive"), ...)
## S3 method for class 'glm'
rstandard(model, infl = influence(model, do.coef = FALSE),
           type = c("deviance", "pearson"), ...)

rstudent(model, ...)
## S3 method for class 'lm'
rstudent(model, infl = lm.influence(model, do.coef = FALSE),
          res = infl$wt.res, ...)
## S3 method for class 'glm'
rstudent(model, infl = influence(model, do.coef = FALSE), ...)

dffits(model, infl = , res = )

dfbeta(model, ...)
## S3 method for class 'lm'
dfbeta(model, infl = lm.influence(model, do.coef = TRUE), ...)

dfbetas(model, ...)
## S3 method for class 'lm'
dfbetas(model, infl = lm.influence(model, do.coef = TRUE), ...)

covratio(model, infl = lm.influence(model, do.coef = FALSE),
          res = weighted.residuals(model))

cooks.distance(model, ...)
## S3 method for class 'lm'
cooks.distance(model, infl = lm.influence(model, do.coef = FALSE),
               res = weighted.residuals(model),
               sd = sqrt(deviance(model)/df.residual(model)),
               hat = infl$hat, ...)
## S3 method for class 'glm'
cooks.distance(model, infl = influence(model, do.coef = FALSE),
               res = infl$pear.res,
               dispersion = summary(model)$dispersion,
               hat = infl$hat, ...)

hatvalues(model, ...)
```

```
## S3 method for class 'lm'
hatvalues(model, infl = lm.influence(model, do.coef = FALSE), ...)

hat(x, intercept = TRUE)
```

```
model          lmglm
infl           lm.influenceinfluenceglmstudentcooks.distance
res
sd
dispersion     glm
hat             $H_{ii}$ 
type           rstandardlmglm
x               $X$ 
intercept      x
...
```

```
influence.measures"infl"
dfbetasdffitscovratiocooks.distancecstandardrstudent
lm()"mlm"
 $F$ 
inflressdml.influenceinfluence
weights == 0na.action = na.exclude
rstandard(*, type = "predictive")model

PRESS <- sum(rstandard(model, type="pred")^2)

hat()hatvalues()
```

```
hatvaluesdfbetadfbetas
```

```
car
```

<https://www.john-fox.ca/Companion/>

[influence](#)[lm.influence](#)

hat

```
require(graphics)
```

```
## Analysis of the life-cycle savings data
## given in Belsley, Kuh and Welsch.
lm.SR <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)

inflm.SR <- influence.measures(lm.SR)
which(apply(inflm.SR$is.inf, 1, any))
# which observations 'are' influential
summary(inflm.SR) # only these
inflm.SR          # all
plot(rstudent(lm.SR) ~ hatvalues(lm.SR)) # recommended by some
plot(lm.SR, which = 5) # an enhanced version of that via plot(<lm>)

## The 'infl' argument is not needed, but avoids recomputation:
rs <- rstandard(lm.SR)
iflSR <- influence(lm.SR)
all.equal(rs, rstandard(lm.SR, infl = iflSR), tolerance = 1e-10)
## to "see" the larger values:
1000 * round(dfbetas(lm.SR, infl = iflSR), 3)
cat("PRESS :"); (PRESS <- sum( rstandard(lm.SR, type = "predictive")^2 ))
stopifnot(all.equal(PRESS, sum( (residuals(lm.SR) / (1 - iflSR$hat))^2)))

## Show that "PRE-residuals" == L.O.O. Crossvalidation (CV) errors:
X <- model.matrix(lm.SR)
y <- model.response(model.frame(lm.SR))
## Leave-one-out CV least-squares prediction errors (relatively fast)
rCV <- vapply(seq_len(nrow(X)), function(i)
              y[i] - X[i,] %*% .lm.fit(X[-i,], y[-i])$coefficients,
              numeric(1))
## are the same as the *faster* rstandard(*, "pred") :
stopifnot(all.equal(rCV, unname(rstandard(lm.SR, type = "predictive"))))

## Huber's data [Atkinson 1985]
xh <- c(-4:0, 10)
yh <- c(2.48, .73, -.04, -1.44, -1.32, 0)
lmH <- lm(yh ~ xh)
summary(lmH)
im <- influence.measures(lmH)
im
is.inf <- apply(im$is.inf, 1, any)
plot(xh,yh, main = "Huber's data: L.S. line and influential obs.")
abline(lmH); points(xh[is.inf], yh[is.inf], pch = 20, col = 2)

## Irwin's data [Williams 1987]
xi <- 1:5
yi <- c(0,2,14,19,30) # number of mice responding to dose xi
mi <- rep(40, 5)      # number of mice exposed
glmI <- glm(cbind(yi, mi -yi) ~ xi, family = binomial)
summary(glmI)
```

```

signif(cooks.distance(glmI), 3) # ~= Ci in Table 3, p.184
imI <- influence.measures(glmI)
  imI
stopifnot(all.equal(imI$infmat[, "cook.d"],
  cooks.distance(glmI)))

```

integrate

```

integrate(f, lower, upper, ..., subdivisions = 100L,
  rel.tol = .Machine$double.eps^0.25, abs.tol = rel.tol,
  stop.on.error = TRUE, keep.xy = FALSE, aux = NULL)

```

```

f
lowerupper
...          f
subdivisions
rel.tol
abs.tol
stop.on.error  message
keep.xy
aux

```

```
...
```

```

rel.tolmax(50*.Machine$double.eps, 0.5e-28)abs.tol <= 0
/src/appl/integrate.cier >= 1
≤lowerupper

```

```

"integrate"
value
abs.error
subdivisions
message      "OK"
call

```


fVectorizef

dqagsdqagi

```
integrate(dnorm, -1.96, 1.96)
integrate(dnorm, -Inf, Inf)

## a slowly-convergent integral
integrand <- function(x) {1/((x+1)*sqrt(x))}
integrate(integrand, lower = 0, upper = Inf)

## don't do this if you really want the integral from 0 to Inf
integrate(integrand, lower = 0, upper = 10)
integrate(integrand, lower = 0, upper = 100000)
integrate(integrand, lower = 0, upper = 1000000, stop.on.error = FALSE)

## some functions do not handle vector input properly
f <- function(x) 2.0
try(integrate(f, 0, 1))
integrate(Vectorize(f), 0, 1) ## correct
integrate(function(x) rep(2.0, length(x)), 0, 1) ## correct

## integrate can fail if misused
integrate(dnorm, 0, 2)
integrate(dnorm, 0, 20)
integrate(dnorm, 0, 200)
integrate(dnorm, 0, 2000)
integrate(dnorm, 0, 20000) ## "fails" on many systems -- "wrongly" giving '0'
integrate(dnorm, 0, Inf) ## works

integrate(dnorm, 0:1, 20) #-> error!
## "silently" gave integrate(dnorm, 0, 20) in earlier versions of R
```

interaction.plot

```

interaction.plot(x.factor, trace.factor, response, fun = mean,
                 type = c("l", "p", "b", "o", "c"), legend = TRUE,
                 trace.label = deparse1(substitute(trace.factor)),
                 fixed = FALSE,
                 xlab = deparse1(substitute(x.factor)),
                 ylab = ylabel,
                 ylim = range(cells, na.rm = TRUE),
                 lty = nc:1, col = 1, pch = c(1:9, 0, letters),
                 xpd = NULL, leg.bg = par("bg"), leg.bty = "n",
                 xtick = FALSE, xaxt = par("xaxt"), axes = TRUE,
                 ...)

```

```

x.factor
trace.factor
response
fun
type          plot.default
legend
trace.label
fixed          trace.factorTRUEFALSE
xlabylab
ylim
lty
col
pch
xpd            legendpar(xpd)
leg.bgleg.bty legend()
xtick
xaxtaxes...

```

```

x.factorx.factor

```

```

xlabylabyylimltycolpchxlimxaxs

```

```

require(graphics)

with(ToothGrowth, {
  interaction.plot(dose, supp, len, fixed = TRUE)
  dose <- ordered(dose)
  interaction.plot(dose, supp, len, fixed = TRUE,
                  col = 2:3, leg.bty = "o", xtick = TRUE)
  interaction.plot(dose, supp, len, fixed = TRUE, col = 2:3, type = "p")
})

with(OrchardSprays, {
  interaction.plot(treatment, rowpos, decrease)
  interaction.plot(rowpos, treatment, decrease, cex.axis = 0.8)
  ## order the rows by their mean effect
  rowpos <- factor(rowpos,
                  levels = sort.list(tapply(decrease, rowpos, mean)))
  interaction.plot(rowpos, treatment, decrease, col = 2:9, lty = 1)
})

```

IQR

x

IQR(x, na.rm = FALSE, type = 7)

x

na.rm

type [quantile](#)

[quantile](#)IQR(x) = quantile(x, 3/4) - quantile(x, 1/4)

$N(m, 1)X$ IQR(X) $2 \cdot \text{qnorm}(3/4) = 1.3490$ IQR(x) / 1.349

[fivenummadrangequantile](#)

IQR(rivers)

`is.empty.model`

`is.empty.model()`

`is.empty.model(x)`

`x` `termsterms`

`TRUE`

`lmglm`

```
y <- rnorm(20)
is.empty.model(y ~ 0)
is.empty.model(y ~ -1)
is.empty.model(lm(y ~ 0))
```

`isoreg`

`isoreg(x, y = NULL)`

`xy` `xy.coordssum(y)`

$m(x)$ `cumsum(y)` $m'(x)$ $m(x)$
`as.stepfun()``stepfun`

`isoreg()``isoreg`

`x` `x`

`y`

`yf`

`yc`

`iKnots`

`isOrd`

`ord` `if(!isOrd)order(x)x`

`call` `callisoreg()`

iKnots2³¹

`plot.isoregisoMDS()`

```
require(graphics)

(ir <- isoreg(c(1,0,4,3,3,5,4,2,0)))
plot(ir, plot.type = "row")

(ir3 <- isoreg(y3 <- c(1,0,4,3,3,5,4,2, 3))) # last "3", not "0"
(fi3 <- as.stepfun(ir3))
(ir4 <- isoreg(1:10, y4 <- c(5, 9, 1:2, 5:8, 3, 8)))
cat(sprintf("R^2 = %.2f\n",
            1 - sum(residuals(ir4)^2) / ((10-1)*var(y4))))

## If you are interested in the knots alone :
with(ir4, cbind(iKnots, yf[iKnots]))

## Example of unordered x[] with ties:
x <- sample((0:30)/8)
y <- exp(x)
x. <- round(x) # ties!
plot(m <- isoreg(x., y))
stopifnot(all.equal(with(m, yf[iKnots]),
                    as.vector(tapply(y, x., mean)))))
```

KalmanLike

```
KalmanLike(y, mod, nit = 0L, update = FALSE)
KalmanRun(y, mod, nit = 0L, update = FALSE)
KalmanSmooth(y, mod, nit = 0L)
KalmanForecast(n.ahead = 10L, mod, update = FALSE)

makeARIMA(phi, theta, Delta, kappa = 1e6,
          SSinit = c("Gardner1980", "Rossignol2011"),
          tol = .Machine$double.eps)
```

```

y
mod
nit          nit = 0LPn
update       TRUEmod"mod"
n.ahead
phitheta     ≥ 0
Delta        y[t] - Delta[1]*y[t-1] - ...
kappa
SSinit       Pn
tol          solve.defaultSSinit = "Rossignol2011"

```

$aa \leftarrow T a + R e e \sim \mathcal{N}(0, \kappa Q) y = Z' a + \eta (eta \equiv \eta), \eta \sim \mathcal{N}(0, \kappa h) \kappa$

```

T
Z
h
V RQR'
a
P Q
Pn t - 1QKalmanForecast
KalmanSmoothtsSmooth
makeARIMAarima
SSinit = "Gardner1980"SSinitarima()"Rossignol2011"(X_{t-1}, ..., X_{t-p}, Z_t, ..., Z_{t-q})

```

```

KalmanLikeLiks2κ
KalmanRunvaluesKalmanLikereresidstates
KalmanSmoothsmoothnpvarnpp
KalmanForecastpredvars2
makeARIMA

```

https://bugs.r-project.org/show_bug.cgi?id=14682

arimaStructTstsSmooth

```
## an ARIMA fit
fit3 <- arima(presidents, c(3, 0, 0))
predict(fit3, 12)
## reconstruct this
pr <- KalmanForecast(12, fit3$model)
pr$pred + fit3$coef[4]
sqrt(pr$var * fit3$sigma2)
## and now do it year by year
mod <- fit3$model
for(y in 1:3) {
  pr <- KalmanForecast(4, mod, TRUE)
  print(list(pred = pr$pred + fit3$coef["intercept"],
            se = sqrt(pr$var * fit3$sigma2)))
  mod <- attr(pr, "mod")
}
```

kernapply

kernapply

kernapply(x, ...)

```
## Default S3 method:
kernapply(x, k, circular = FALSE, ...)
## S3 method for class 'ts'
kernapply(x, k, circular = FALSE, ...)
## S3 method for class 'vector'
kernapply(x, k, circular = FALSE, ...)

## S3 method for class 'tskernel'
kernapply(x, k, ...)
```

```
x
k          "tskernel"
circular
...
```

fftNROW(x)

[kernelconvolvefilterspectrum](#)

see 'kernel' for examples

kernel

"tskernel"

[printplot\[](#)

kernel(coef, m = 2, r, name)

df.kernel(k)

bandwidth.kernel(k)

is.tskernel(k)

S3 method for class 'tskernel'

plot(x, type = "h", xlab = "k", ylab = "W[k]",
main = attr(x,"name"), ...)

coef "daniell""dirichlet""fejer""modified.daniell"

m coefmm[j]j in 1:length(m)"*daniell"

name

r

kx "tskernel"

typexlabylabmain...

[plot.default](#)

kernel

[\[\(-m\) : mk <- kernel\(*\)sum\(k\[-k\\$m : k\\$m \]\)](#)

df.kernelbandwidth.kernel

kernel()"tskernel"coefm"name"

kernapply

```
require(graphics)

## Demonstrate a simple trading strategy for the
## financial time series German stock index DAX.
x <- EuStockMarkets[,1]
k1 <- kernel("daniell", 50) # a long moving average
k2 <- kernel("daniell", 10) # and a short one
plot(k1)
plot(k2)
x1 <- kernapply(x, k1)
x2 <- kernapply(x, k2)
plot(x)
lines(x1, col = "red") # go long if the short crosses the long upwards
lines(x2, col = "green") # and go short otherwise

## More interesting kernels
kd <- kernel("daniell", c(3, 3))
kd # note the unusual indexing
kd[-2:2]
plot(kernel("fejer", 100, r = 6))
plot(kernel("modified.daniell", c(7,5,3)))

# Reproduce example 10.4.3 from Brockwell and Davis (1991)
spectrum(sunspot.year, kernel = kernel("daniell", c(11,7,3)), log = "no")
```

kmeans

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace = FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)
```

```
x
centers       $k \times$ 
iter.max
nstart       centers
```

```

algorithm      "Lloyd""Forgy"
object         "kmeans"obob <- kmeans(..)
method         "centers"fitted"classes"fitted
trace          "Hartigan-Wong"
...

```

```

 $xkk$ 
knstart> lxifault = 4
 $k$  = lwithinss
 $k$ 

```

```

kmeans"kmeans"printfitted

cluster        1:k
centers
totss
withinss
tot.withinss   sum(withinss)
betweenss      totss-tot.withinss
size
iter
ifault

```

```

require(graphics)

# a 2-dimensional example
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
            matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
(cl <- kmeans(x, 2))
plot(x, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 8, cex = 2)

# sum of squares
ss <- function(x) sum(scale(x, scale = FALSE)^2)

```

```

## cluster centers "fitted" to each obs.:
fitted.x <- fitted(cl); head(fitted.x)
resid.x <- x - fitted(cl)

## Equalities : -----
cbind(cl[c("betweenss", "tot.withinss", "totss")], # the same two columns
      c(ss(fitted.x), ss(resid.x), ss(x)))
stopifnot(all.equal(cl$ totss, ss(x)),
          all.equal(cl$ tot.withinss, ss(resid.x)),
          ## these three are the same:
          all.equal(cl$ betweenss, ss(fitted.x)),
          all.equal(cl$ betweenss, cl$totss - cl$tot.withinss),
          ## and hence also
          all.equal(ss(x), ss(fitted.x) + ss(resid.x))
          )

kmeans(x,1)$withinss # trivial one-cluster, (its W.SS == ss(x))

## random starts do help here with too many clusters
## (and are often recommended anyway!):
## The ordering of the clusters may be platform-dependent.

(cl <- kmeans(x, 5, nstart = 25))

plot(x, col = cl$cluster)
points(cl$centers, col = 1:5, pch = 8)

```

kruskal.test

```

kruskal.test(x, ...)

## Default S3 method:
kruskal.test(x, g, ...)

## S3 method for class 'formula'
kruskal.test(formula, data, subset, na.action, ...)

x
g          xx
formula    response ~ groupresponsegroup
data       model.frameformulaenvironment(formula)
subset
na.action  NAgetOption("na.action")
...

```

```

kruskal.testx
xgkruskal.test(x)kruskal.test(list(x, ...))
xgxx

"htest"

statistic
parameter
p.value
method          "Kruskal-Wallis rank sum test"
data.name

```

```

wilcox.testlmanovat.test
wilcox_test

```

```

## Hollander & Wolfe (1973), 116.
## Mucociliary efficiency from the rate of removal of dust in normal
## subjects, subjects with obstructive airway disease, and subjects
## with asbestosis.
x <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
y <- c(3.8, 2.7, 4.0, 2.4)      # with obstructive airway disease
z <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis
kruskal.test(list(x, y, z))
## Equivalently,
x <- c(x, y, z)
g <- factor(rep(1:3, c(5, 4, 5)),
            labels = c("Normal subjects",
                       "Subjects with obstructive airway disease",
                       "Subjects with asbestosis"))
kruskal.test(x, g)

## Formula interface.
require(graphics)
boxplot(Ozone ~ Month, data = airquality)
kruskal.test(Ozone ~ Month, data = airquality)

```

ks.test

```
ks.test(x, ...)
## Default S3 method:
ks.test(x, y, ...,
        alternative = c("two.sided", "less", "greater"),
        exact = NULL, simulate.p.value = FALSE, B = 2000)
## S3 method for class 'formula'
ks.test(formula, data, subset, na.action, ...)
```

```
x
y          pnorm
...        y
alternative "two.sided""less""greater"
exact      NULLNULL
simulate.p.value
```

```
B
formula      lhs ~ rhslhsrhs1
data         model.frameformulaenvironment(formula)
subset
na.action    NAgetOption("na.action")
```

```
xy
xy...
xy
"two.sided""less""greater"alternativexy"greater" $D^+ = \max_u [F_x(u) - F_y(u)]$ 
alternative = "greater"xyxyt.testwilcox.test
exact = NULLsimulate.p.valueTRUE
...ks.test
```

```
"ks.test""htest"
```

```
statistic
p.value
alternative
method
data.name
```

<https://arxiv.org/abs/2102.08037>

[psmirnov](#)

[shapiro.test](#)

```
require("graphics")

x <- rnorm(50)
y <- runif(30)
# Do x and y come from the same distribution?
ks.test(x, y)
# Does x come from a shifted gamma distribution with shape 3 and rate 2?
ks.test(x+2, "pgamma", 3, 2) # two-sided, exact
ks.test(x+2, "pgamma", 3, 2, exact = FALSE)
ks.test(x+2, "pgamma", 3, 2, alternative = "gr")

# test if x is stochastically larger than x2
x2 <- rnorm(50, -1)
plot(ecdf(x), xlim = range(c(x, x2)))
plot(ecdf(x2), add = TRUE, lty = "dashed")
t.test(x, x2, alternative = "g")
wilcox.test(x, x2, alternative = "g")
ks.test(x, x2, alternative = "l")

# with ties, example from Schröder and Trenkler (1995)
# D = 3/7, p = 8/33 = 0.242424..
ks.test(c(1, 2, 2, 3, 3),
        c(1, 2, 3, 3, 4, 5, 6))# -> exact

# formula interface, see ?wilcox.test
ks.test(Ozone ~ Month, data = airquality,
        subset = Month %in% c(5, 8))
```

ksmooth

```
ksmooth(x, y, kernel = c("box", "normal"), bandwidth = 0.5,  
        range.x = range(x),  
        n.points = max(100L, length(x)), x.points)
```

```
x  
y  
kernel  
bandwidth      ±0.25*bandwidth  
range.x  
n.points  
x.points       n.pointsrange.x
```

```
x  
y          x
```

```
require(graphics)  
  
with(cars, {  
  plot(speed, dist)  
  lines(ksmooth(speed, dist, "normal", bandwidth = 2), col = 2)  
  lines(ksmooth(speed, dist, "normal", bandwidth = 5), col = 3)  
})
```

lag

lag

lag(x, ...)

Default S3 method:

lag(x, k = 1, ...)

x

k

...

[xtsphasTsp](#)

x

kk

[diffdeltat](#)

lag(ldeaths, 12) # starts one year earlier

lag.plot

```
lag.plot(x, lags = 1, layout = NULL, set.lags = 1:lags,  
         main = NULL, asp = 1,  
         diag = TRUE, diag.col = "gray", type = "p", oma = NULL,  
         ask = NULL, do.lines = (n <= 150), labels = do.lines,  
         ...)
```

x	
lags	set.lags
layout	mfrowpar()n2mfrow
set.lags	1:lags
main	
asp	plot.default
diag	
diag.col	if(diag)
type	plot.ts
oma	par
ask	NULL
do.lines	
labels	
...	plot.tsxlabylabmgpcol.labfont.labxy.labelsxy.lines

```
par(mfrow)  
ask = NULLpar(ask = TRUE)
```

```
main =head =
```

[plot.ts](#)

```

require(graphics)

lag.plot(nhtemp, 8, diag.col = "forest green")
lag.plot(nhtemp, 5, main = "Average Temperatures in New Haven")
## ask defaults to TRUE when we have more than one page:
lag.plot(nhtemp, 6, layout = c(2,1), asp = NA,
        main = "New Haven Temperatures", col.main = "blue")

## Multivariate (but non-stationary! ...)
lag.plot(freeny.x, lags = 3)

## no lines for long series :
lag.plot(sqrt(sunspots), set.lags = c(1:4, 9:12), pch = ".", col = "gold")

```

line

```

iter = 1

```

```

line(x, y, iter = 1)

```

```

xy          xy.coords
iter        1

```

```

nn %% 3line()x[.] <= q1x[.] >= q2q1, q2p = 1/3p = 2/3(x[j1]+x[j2])/2j1 =
floor(p*(n-1))j2 = ceiling(p*(n-1))n = length(x)

```

```

"tukeyline"
coefresidualsfittedprint

```

yx

```

lm
rlm()lmrob()

```

```

require(graphics)

plot(cars)
(z <- line(cars))
abline(coef(z))
## Tukey-Anscombe Plot :
plot(residuals(z) ~ fitted(z), main = deparse(z$call))

## Andrew Siegel's pathological 9-point data, y-values multiplied by 3:
d.AS <- data.frame(x = c(-4:3, 12), y = 3*c(rep(0,6), -5, 5, 1))
cAS <- with(d.AS, t(sapply(1:10,
                           function(it) line(x,y, iter=it)$coefficients)))
dimnames(cAS) <- list(paste("it =", format(1:10)), c("intercept", "slope"))
cAS
## iterations started to oscillate, repeating iteration 7,8 indefinitely

```

listof

```

"listof" aov "lm" alias
coef [print]

```

lm

[lm](#) [aov](#)

```

lm(formula, data, subset, weights, na.action,
    method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
    singular.ok = TRUE, contrasts = NULL, offset, ...)

```

```

## S3 method for class 'lm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

```

formula	"formula"
data	as.data.frame dataenvironment (formula) lm
subset	
weights	NULL weightssum (w*e^2)
na.action	NA na.actionoptions na.fail na.omit NULL na.exclude
method	method = "qr"method = "model.frame"model = TRUE
modelxyqr	TRUE
singular.ok	FALSE
contrasts	contrasts.arg model.matrix.default
offset	NULL offsetmodel.offset
...	lm()
digits	format (coef (x), .) print()

```
lmresponse ~ termsresponsetermsresponsefirst + secondfirstsecondfirst:second
firstsecondfirst*secondfirstsecondfirst + second + first:second
```

```
offset
```

```
response"mlm"
```

```
model.matrixtermsaovdemo(glm.vr)
```

```
y ~ x - 1y ~ 0 + xformula
```

```
NULLweightsweightsweights $w_i y_i w_i w_i y_i$ 
```

```
lmlm.fit
```

```
weightssubsetoffsetformuladataformulamean(x)subsetmodel.frame
```

```
lmclass"lm"c("mlm", "lm")
```

```
summaryanovacoefficientseffectsfitted.valuesresidualslm
```

```
"lm"
```

```
coefficients
```

```
residuals
```

```
fitted.values
```

```
rank
```

```
weights
```

```
df.residual
```

```
call
```

```
terms          terms
```

```
contrasts
```

```
xlevels
```

```
offset
```

```
y
```

```
x
```

```
model
```

```
na.action      model.frameNA
```

```
assigneffectsqrsummaryeffects
```

```
lm
```

```
na.action = NULLNANA
```

```
datats.intersect(..., dframe = TRUE)na.actionlmna.action = NULL
```

```
summary.lmanova.lmaov
coefeffectsresidualsfittedvcov
predict.lmpredictconfint
lm.influenceglm
lm.fitlm.wfit
lm()anscombeattitudedefreenyLifeCycleSavingslongleystacklossswiss
biglm
```

```
require(graphics)
```

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept

anova(lm.D9)
summary(lm.D90)

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

### less simple examples in "See Also" above
```

lm.fit

```
lm
lmfit()
.lm.fit()lm.fit()glm.fit()lsfit()
```

```
lm.fit(x, y, offset = NULL, method = "qr", tol = 1e-7,
       singular.ok = TRUE, ...)
```

```
lm.wfit(x, y, w, offset = NULL, method = "qr", tol = 1e-7,
       singular.ok = TRUE, ...)
```

```
.lm.fit(x, y, tol = 1e-7)
```

```

x          n * p
y          nn
w          nwfitwsum(w * e^2)
offset     n
method     method = "qr"
tol        qr
singular.ok FALSE
...

```

```
yoffsety
```

```
listlm.fitlm.wfit
```

```

coefficients p
residuals    n
fitted.values n
effects      nrank
weights      n*wfit*
rank
df.residual
qr           qr
effectsqr
.lm.fit()qrpivoted

```

```
lm
```

```

require(utils)
set.seed(129)

n <- 7 ; p <- 2
X <- matrix(rnorm(n * p), n, p) # no intercept!
y <- rnorm(n)
w <- rnorm(n)^2

str(lmw <- lm.wfit(x = X, y = y, w = w))

str(lm. <- lm.fit (x = X, y = y))

## fits w/o intercept:
all.equal(unname(coef(lm(y ~ X-1))),
          unname(coef( lm.fit(X,y))))
all.equal(unname(coef( lm.fit(X,y))),

```

```

      coef(.lm.fit(X,y)))

if(require("microbenchmark")) {
  mb <- microbenchmark(lm(y~X-1), lm.fit(X,y), .lm.fit(X,y))
  print(mb)
  boxplot(mb, notch=TRUE)
}

```

lm.influence

```

influence(model, ...)
## S3 method for class 'lm'
influence(model, do.coef = TRUE, ...)
## S3 method for class 'glm'
influence(model, do.coef = TRUE, ...)

lm.influence(model, do.coef = TRUE)

qr.influence(qr, res, tol = 10 * .Machine$double.eps)

```

```

model          lmglm
do.coef        coefficients  $O(n^2p)$ 
...
qr             qr\(\)list"qr"
res
tol

```

```

influence.measures\(\)lm.influence
sigmacoefficientsNaN
naresidNaNa.action = na.exclude
qr.influence()lm.influence(*, doc.coef = FALSE)

```

```

na.actionna.exclude

```

```

hat
coefficients   do.coef
sigma          NaN
wt.res         glm

qr.influence()hatsigmanames

```

```

coefficientslm.influence
O(np2)do.coef = FALSE
weights == 0
na.action = na.excludena.exclude

```

[influence.measures](#)

[summary.lmsummary](#)
[influence.measures](#)
[hat](#)
[dfbetasdffitscovratiocooks.distance](#)[lm](#)

```

## Analysis of the life-cycle savings data
## given in Belsley, Kuh and Welsch.
summary(lm.SR <- lm(sr ~ pop15 + pop75 + dpi + ddpi,
                    data = LifeCycleSavings),
        correlation = TRUE)
utils::str(lmI <- lm.influence(lm.SR))

qRes <- qr(lm.SR) # == lm.SR $ qr
qrI <- qr.influence(qRes, residuals(lm.SR))
strip <- function(x) lapply(lapply(x, unname), drop)
stopifnot(identical(strip(qrI),
                      strip(lmI[c("hat", "sigma")]))))

## For more "user level" examples, use example(influence.measures)

```

lm.summaries

[methods](#)"lm"

```

## S3 method for class 'lm'
family(object, ...)

## S3 method for class 'lm'
formula(x, ...)

## S3 method for class 'lm'
residuals(object,
           type = c("working", "response", "deviance", "pearson",
                    "partial"),
           ...)

## S3 method for class 'lm'
labels(object, ...)

```



```

objectx      lmlmaov
...
type

coefeffectsfittedresidualslm
rstudentweighted.residuals
residualsna.actionna.action = na.omitna.action = na.excludeNAnaresid
"lm"labels

lmanova.lm
coefdeviancedf.residualeffectsfittedglminfluenceweighted.residualsresiduals
residuals.glmsummary.lmweights
rstandardrstudent

##-- Continuing the lm(.) example:
coef(lm.D90) # the bare coefficients

## The 2 basic regression diagnostic plots [plot.lm(.) is preferred]
plot(resid(lm.D90), fitted(lm.D90)) # Tukey-Anscombe's
abline(h = 0, lty = 2, col = "gray")

qqnorm(residuals(lm.D90))

```

loadings

`factanal()`

```

loadings(x, ...)

## S3 method for class 'loadings'
print(x, digits = 3, cutoff = 0.1, sort = FALSE, ...)

## S3 method for class 'factanal'
print(x, digits = 3, ...)

```

```

x          "factanal""princomp"loadings
digits
cutoff
sort
...        factanalprint(<loadings>, ...)cutoff = <frac>loadings

```

princomp

```
print"factanal""loadings"cutoffsort
```

loadings...

factanalprincomp

loess

```

loess(formula, data, weights, subset, na.action, model = FALSE,
      span = 0.75, enp.target, degree = 2,
      parametric = FALSE, drop.square = FALSE, normalize = TRUE,
      family = c("gaussian", "symmetric"),
      method = c("loess", "model.frame"),
      control = loess.control(...), ...)

```

```

formula
data          as.data.frame(dataenvironment(formula))loess
weights
subset
na.action     getOption("na.action")
model
span           $\alpha$ 
enp.target    span
degree
parametric
drop.square   degree = 2parametric
normalize
family        "gaussian""symmetric"
method
control       loess.control
...           control

```

```

xxxspanenp.target $\alpha < 1\alpha(1 - (\text{dist}/\text{maxdist})^3)^3\alpha > 1\alpha^{1/p}p$ 
family="symmetric"
loess.control

```

```

"loess"print()summary()predictanova

```

```

cloessloess
loess
degree = 0

```

```

cloess

```

```

cloessdloesshttps://netlib.org/a/

```

```

loess.controlpredict.loess
lowessloess

```

```

cars.lo <- loess(dist ~ speed, cars)
predict(cars.lo, data.frame(speed = seq(5, 30, 1)), se = TRUE)
# to allow extrapolation
cars.lo2 <- loess(dist ~ speed, cars,
                  control = loess.control(surface = "direct"))
predict(cars.lo2, data.frame(speed = seq(5, 30, 1)), se = TRUE)

```

loess.control	loess
---------------	-------

```

loess

```

```

loess.control(surface = c("interpolate", "direct"),
               statistics = c("approximate", "exact", "none"),
               trace.hat = c("exact", "approximate"),
               cell = 0.2, iterations = 4, iterTrace = FALSE, ...)

```

```

surface      "direct"
statistics
trace.hat    (surface = "interpolate", statistics = "approximate")
cell         floor(n*span*cell)
iterations   family"symmetric"
iterTrace    iterations $\geq$  2
...

```

```

surface
statistics
trace.hat
cell
iterations
iterTrace

```

loess

Logistic

locationscale

```

dlogis(x, location = 0, scale = 1, log = FALSE)
plogis(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qlogis(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rlogis(n, location = 0, scale = 1)

```

```

xq
p
n          length(n) > 1
locationscale
loglog.p
lower.tail  $P[X \leq x]P[X > x]$ 

```

locationscale01

location= μ scale= σ

$$F(x) = \frac{1}{1 + e^{-(x-\mu)/\sigma}}$$

$$f(x) = \frac{1}{\sigma} \frac{e^{(x-\mu)/\sigma}}{(1 + e^{(x-\mu)/\sigma})^2}$$

$$\mu\pi^2/3\sigma^2$$

dlogisplogisqlogisrlogis

nrlogis

n

qlogis(p)*logit*(p) = log p/(1 - p)plogis(x)

plogis(x) == (1+ tanh(x/2))/2

[dpq]logis

rlogis

var(rlogis(4000, 0, scale = 5)) # approximately (+/- 3)
 $\pi^2/3 * 5^2$

logLik

"glm""lm""nls""Arima""fitdistr""negbin""polr""multinom""glis""gnls""lme"

logLik(object, ...)

S3 method for class 'lm'

logLik(object, REML = FALSE, ...)

object

...

REML TRUEFALSEFALSE

logLikAIC"lme"

"glm"familyaic()gaussianGammainverse.gaussian"glm"

"lm"

logLik"df"

print"logLik"

"nobs"REML = TRUE

logLik.lm

logLik.glslogLik.lme

AIC

x <- 1:5

lmx <- lm(x ~ 1)

logLik(lmx) # using print.logLik() method

utils::str(logLik(lmx))

lm method

(fm1 <- lm(rating ~ ., data = attitude))

logLik(fm1)

logLik(fm1, REML = TRUE)

utils::data(Orthodont, package = "nlme")

fm1 <- lm(distance ~ Sex * age, Orthodont)

logLik(fm1)

logLik(fm1, REML = TRUE)

loglin

loglin

```
loglin(table, margin, start = rep(1, length(table)), fit = FALSE,  
       eps = 0.1, iter = 20, param = FALSE, print = TRUE)
```

table	table
margin	 list(c(1, 2), c(1, 3)) names(dimnames(table))NULLlist()
start	tablestart
fit	
eps	
iter	
param	
print	TRUE

itereps
df
marginmargin
loglmloglinlmglm

lrt	
pearson	
df	
margin	marginNULL
fit	tablefitTRUE
param	paramTRUE

```
table
loglm
glm
```

```
## Model of joint independence of sex from hair and eye color.
fm <- loglin(HairEyeColor, list(c(1, 2), c(1, 3), c(2, 3)))
fm
1 - pchisq(fm$lrt, fm$df)
## Model with no three-factor interactions fits well.
```

Lognormal

meanlogsdlog

```
dlnorm(x, meanlog = 0, sdlog = 1, log = FALSE)
plnorm(q, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
qlnorm(p, meanlog = 0, sdlog = 1, lower.tail = TRUE, log.p = FALSE)
rlnorm(n, meanlog = 0, sdlog = 1)
```

```
xq
p
n          length(n) > 1
meanlogsdlog 01
loglog.p
lower.tail     $P[X \leq x]P[X > x]$ 
```

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

$$\frac{\mu\sigma E(X)}{\sqrt{\exp(\sigma^2)-1}\sigma} = \frac{\exp(\mu + 1/2\sigma^2)med(X)}{\sqrt{\exp(\sigma^2)-1}\sigma} = \frac{\exp(\mu)Var(X)}{\sqrt{\exp(\sigma^2)-1}\sigma} = \frac{\exp(2\mu + \sigma^2)(\exp(\sigma^2) - 1)}{\sqrt{\exp(\sigma^2)-1}\sigma} < 1/2$$

```
dlnormplnormqlnormrlnorm
nrlnorm
n
```

$$H(t) = -\log(1 - F(t)) - \text{plnorm}(t, r, \text{lower} = \text{FALSE}, \text{log} = \text{TRUE})$$


```
dlnorm[pqr]lnorm
exp(meanlog)sdlog = 0
```

[dnorm](#)

```
dlnorm(1) == dnorm(0)
```

lowess

```
lowess(x, y = NULL, f = 2/3, iter = 3, delta = 0.01 * diff(range(x)))
```

xy	xy.coords
f	
iter	iterlowess
delta	x

```
lowesssrc/library/stats/src/lowess.docfloor(f*n)x
iter > 0x
deltadelta
```

```
lowessxylines
```

[loess](#)lowess

```
require(graphics)

plot(cars, main = "lowess(cars)")
lines(lowess(cars), col = 2)
lines(lowess(cars, f = .2), col = 3)
legend(5, 120, c(paste("f = ", c("2/3", ".2"))), lty = 1, col = 2:3)
```

ls.diag

lsfit

ls.diag(ls.out)

ls.out `lsfit()`

list

std.dev σ

hat $h_{ii}H$

std.res

stud.res

cooks

dfits

correlation

std.err

cov.scaled

cov.unscaled

`hatls.printlm.influencesummary.lmanova`

```
##-- Using the same data as the lm(.) example:
lsD9 <- lsfit(x = as.numeric(gl(2, 10, 20)), y = weight)
dlsD9 <- ls.diag(lsD9)
utils::str(dlsD9, give.attr = FALSE)
abs(1 - sum(dlsD9$hat) / 2) < 10*.Machine$double.eps # sum(h.ii) = p
plot(dlsD9$hat, dlsD9$stud.res, xlim = c(0, 0.11))
abline(h = 0, lty = 2, col = "lightgray")
```

ls.print	lsfit
<pre> print.itTRUE ls.print(ls.out, digits = 4, print.it = TRUE) ls.out lsfit() digits print.it summary coef.table summary(lm(...))anova(lm(...)) ls.diaglsfitlm lm.influence </pre>	
lsfit	
β $Y = X\beta + \epsilon$ <pre> lsfit(x, y, wt = NULL, intercept = TRUE, tolerance = 1e-07, yname = NULL) x y wt intercept tolerance yname </pre>	

wt

coef β
residuals
intercept
qr

`lmls.printls.diag`

```
##-- Using the same data as the lm(.) example:  
lsD9 <- lsfit(x = unclass(gl(2, 10)), y = weight)  
ls.print(lsD9)
```

mad

```
mad(x, center = median(x), constant = 1.4826, na.rm = FALSE,  
    low = FALSE, high = FALSE)
```

x
center
constant
na.rm TRUENAx
low TRUE
high TRUE

```
constant * cMedian(abs(x - center))centermedian(x)cMedianlowhigh  
n = 1center0  
constant = 1.48261/ $\Phi^{-1}(\frac{3}{4})$ 1/qnorm(3/4)
```

$$E[{\it mad}(X_1, \dots, X_n)] = \sigma$$

$X_i \sim N(\mu, \sigma^2)$
na.rmTRUENAxNAxmadNA

IQRmedianvar

```
mad(c(1:9))
print(mad(c(1:9), constant = 1)) ==
      mad(c(1:8, 100), constant = 1)      # = 2 ; TRUE
x <- c(1,2,3,5,7,8)
sort(abs(x - median(x)))
c(mad(x, constant = 1),
  mad(x, constant = 1, low = TRUE),
  mad(x, constant = 1, high = TRUE))
```

mahalanobis

$x\mu\text{center}\Sigma\text{cov}x$

$$D^2 = (x - \mu)' \Sigma^{-1} (x - \mu)$$

mahalanobis(x, center, cov, inverted = FALSE, ...)

x	p
center	p FALSE
cov	$p \times p$
inverted	TRUEcov
...	solveinverted

covvar

require(graphics)

```
ma <- cbind(1:6, 1:3)
(S <- var(ma))
mahalanobis(c(0, 0), 1:2, S)
```

```
x <- matrix(rnorm(100*3), ncol = 3)
stopifnot(mahalanobis(x, 0, diag(ncol(x))) == rowSums(x*x))
##- Here, D^2 = usual squared Euclidean distances
```

```
Sx <- cov(x)
D2 <- mahalanobis(x, colMeans(x), Sx)
plot(density(D2, bw = 0.5),
     main="Squared Mahalanobis distances, n=100, p=3") ; rug(D2)
qqplot(qchisq(ppoints(100), df = 3), D2,
      main = expression("Q-Q plot of Mahalanobis" * ~D^2 *
                        " vs. quantiles of" * ~chi[3]^2))
abline(0, 1, col = 'gray')
```

`make.link`

`familyglm()` $d\mu/d\eta$

`make.link(link)`

`link` `"logit""probit""cauchit""cloglog""identity""log""sqrt""1/mu^2"`
 `"inverse"`

`"link-glm"`

`linkfun` `function(mu)`

`linkinv` `function(eta)`

`mu.eta` `function(eta)` $d\mu/d\eta$

`valideta` `function(eta)TRUE``etalinkinv`

`name`

`powerglmfamily`

`utils::str(make.link("logit"))`

`makepredictcall`

`model.frame.default``polyns`

`makepredictcall(var, call)`

`var`

`call`

`polybsnsscalemodel.frame.default``predvars``makepredictcall.ns`

`terms``predvar``term``terms`

callpredvars

[model.framepolyscalebsns](#)

[cars](#)

require(graphics)

```
## using poly: this did not work in R < 1.5.0
fm <- lm(weight ~ poly(height, 2), data = women)
plot(women, xlab = "Height (in)", ylab = "Weight (lb)")
ht <- seq(57, 73, length.out = 200)
nD <- data.frame(height = ht)
pfm <- predict(fm, nD)
lines(ht, pfm)
pf2 <- predict(update(fm, ~ stats::poly(height, 2)), nD)
stopifnot(all.equal(pfm, pf2)) ## was off (rel.diff. 0.0766) in R <= 3.5.0

## see also example(cars)

## see bs and ns for spline examples.
```

manova

manova(...)

... [aov](#)

"manova" "aov" summarymanova[aov](#) "manova"

[aov](#)

[aovsummary.manova](#)

```
## Set orthogonal contrasts.
op <- options(contrasts = c("contr.helmert", "contr.poly"))

## Fake a 2nd response variable
npk2 <- within(npk, foo <- rnorm(24))
( npk2.aov <- manova(cbind(yield, foo) ~ block + N*P*K, npk2) )
summary(npk2.aov)

( npk2.aovE <- manova(cbind(yield, foo) ~ N*P*K + Error(block), npk2) )
summary(npk2.aovE)
```

```
mantelhaen.test
```

```
mantelhaen.test(x, y = NULL, z = NULL,
                 alternative = c("two.sided", "less", "greater"),
                 correct = TRUE, exact = FALSE, conf.level = 0.95)
```

```
x
y      x
z      xyx
alternative  "two.sided""greater""less"K
correct      K
exact      K
conf.level   K
```

```
xNAxyzNA
```

```
"htest"
statistic      K
parameter      1
p.value
conf.int      K
estimate      K
null.value     1K
alternative     K
method
data.name
```


K

```
## Agresti (1990), pages 231--237, Penicillin and Rabbits
## Investigation of the effectiveness of immediately injected or 1.5
## hours delayed penicillin in protecting rabbits against a lethal
## injection with beta-hemolytic streptococci.
Rabbits <-
array(c(0, 0, 6, 5,
        3, 0, 3, 6,
        6, 2, 0, 4,
        5, 6, 1, 0,
        2, 5, 0, 0),
      dim = c(2, 2, 5),
      dimnames = list(
        Delay = c("None", "1.5h"),
        Response = c("Cured", "Died"),
        Penicillin.Level = c("1/8", "1/4", "1/2", "1", "4")))
Rabbits
## Classical Mantel-Haenszel test
mantelhaen.test(Rabbits)
## => p = 0.047, some evidence for higher cure rate of immediate
## injection
## Exact conditional test
mantelhaen.test(Rabbits, exact = TRUE)
## => p = 0.040
## Exact conditional test for one-sided alternative of a higher
## cure rate for immediate injection
mantelhaen.test(Rabbits, exact = TRUE, alternative = "greater")
## => p = 0.020

## UC Berkeley Student Admissions
mantelhaen.test(UCBAdmissions)
## No evidence for association between admission and gender
## when adjusted for department. However,
apply(UCBAdmissions, 3, function(x) (x[1,1]*x[2,2])/(x[1,2]*x[2,1]))
## This suggests that the assumption of homogeneous (conditional)
## odds ratios may be violated. The traditional approach would be
## using the Woolf test for interaction:
woolf <- function(x) {
  x <- x + 1 / 2
  k <- dim(x)[3]
  or <- apply(x, 3, function(x) (x[1,1]*x[2,2])/(x[1,2]*x[2,1]))
  w <- apply(x, 3, function(x) 1 / sum(1 / x))
  1 - pchisq(sum(w * (log(or) - weighted.mean(log(or), w)) ^ 2), k - 1)
}
woolf(UCBAdmissions)
## => p = 0.003, indicating that there is significant heterogeneity.
## (And hence the Mantel-Haenszel test cannot be used.)
```

```
## Agresti (2002), p. 287f and p. 297.
## Job Satisfaction example.
Satisfaction <-
  as.table(array(c(1, 2, 0, 0, 3, 3, 1, 2,
                  11, 17, 8, 4, 2, 3, 5, 2,
                  1, 0, 0, 0, 1, 3, 0, 1,
                  2, 5, 7, 9, 1, 1, 3, 6),
                dim = c(4, 4, 2),
                dimnames =
                  list(Income =
                     c("<5000", "5000-15000",
                       "15000-25000", ">25000"),
                     "Job Satisfaction" =
                     c("V_D", "L_S", "M_S", "V_S"),
                     Gender = c("Female", "Male")))))
## (Satisfaction categories abbreviated for convenience.)
ftable(. ~ Gender + Income, Satisfaction)
## Table 7.8 in Agresti (2002), p. 288.
mantelhaen.test(Satisfaction)
## See Table 7.12 in Agresti (2002), p. 297.
```

mauchly.test

```
mauchly.test(object, ...)
## S3 method for class 'mlm'
mauchly.test(object, ...)
## S3 method for class 'SSD'
mauchly.test(object, Sigma = diag(nrow = p),
              T = Thin.row(Proj(M) - Proj(X)), M = diag(nrow = p), X = ~0,
              idata = data.frame(index = seq_len(p)), ...)
```

```
object      SSDmlm
Sigma
T           MX
M
X
idata
...
```

```
"mlm""SSD"
```

```
SSDmlm
```

```
TTMXidataM/X
```

```
X = ~1
```

```
pprojThin.row
```

"htest"

[SSDanova.mlmrWishart](#)

```
utils::example(SSD) # Brings in the mlmfit and reacttime objects
```

```
### traditional test of intrasubj. contrasts  
mauchly.test(mlmfit, X = ~1)
```

```
### tests using intra-subject 3x2 design  
idata <- data.frame(deg = gl(3, 1, 6, labels = c(0,4,8)),  
                    noise = gl(2, 3, 6, labels = c("A","P")))  
mauchly.test(mlmfit, X = ~ deg + noise, idata = idata)  
mauchly.test(mlmfit, M = ~ deg + noise, X = ~ noise, idata = idata)
```

mcnemar.test

```
mcnemar.test(x, y = NULL, correct = TRUE)
```

```
x  
y          x  
correct
```

```
[i,j][j,i]  
xxy  
correctTRUE
```

```
"htest"  
statistic  
parameter  
p.value  
method  
data.name
```

```
## Agresti (1990), p. 350.
## Presidential Approval Ratings.
## Approval of the President's performance in office in two surveys,
## one month apart, for a random sample of 1600 voting-age Americans.
Performance <-
matrix(c(794, 86, 150, 570),
       nrow = 2,
       dimnames = list("1st Survey" = c("Approve", "Disapprove"),
                        "2nd Survey" = c("Approve", "Disapprove")))
Performance
mcnemar.test(Performance)
## => significant change (in fact, drop) in approval ratings
```

median

```
median(x, na.rm = FALSE, ...)
## Default S3 method:
median(x, na.rm = FALSE, ...)
```

```
x
na.rm      NA
...
```

```
is.nasortmean"Date"
```

```
xx
na.rm = FALSENANaxx[NA_integer_]
```

quantile

```
median(1:4)          # = 2.5 [even number]
median(c(1:3, 100, 1000)) # = 3 [odd, robust]
```

medpolish

```
medpolish(x, eps = 0.01, maxiter = 10, trace.iter = TRUE,  
          na.rm = FALSE)
```

```
x  
eps  
maxiter  
trace.iter  
na.rm
```

```
epsmaxitertrace.iterTRUEna.rmFALSENAxNA  
medpolishmedpolishprintplot
```

```
medpolish  
overall  
row  
col  
residuals  
name
```

[medianaov](#)

```
require(graphics)  
  
## Deaths from sport parachuting; from ABC of EDA, p.224:  
deaths <-  
  rbind(c(14,15,14),  
        c( 7, 4, 7),  
        c( 8, 2,10),  
        c(15, 9,10),  
        c( 0, 2, 0))  
dimnames(deaths) <- list(c("1-24", "25-74", "75-199", "200++", "NA"),  
                        paste(1973:1975))  
  
deaths  
(med.d <- medpolish(deaths))  
plot(med.d)  
## Check decomposition:  
all(deaths ==  
     med.d$overall + outer(med.d$row,med.d$col, `+`) + med.d$residuals)
```

`model.extract`

`model.frame`

```
model.extract(frame, component)
model.offset(x)
model.response(data, type = "any")
model.weights(x)
```

```
framexdata      model.frame
component       "weights" "subset"
type            "any" "numeric" "double" "double"
```

```
model.extractetastartmuststartglm
model.extract(m,      "offset")model.extract(m,      "response")model.offset(m)
model.response(m)model.offsetoffsetoffset
model.weightsmodel.extract(, "weights")
```

```
model.response()"Asis"I(.)
model.offsetNULL
```

`model.frameoffset`

```
a <- model.frame(cbind(ncases,ncontrols) ~ agegp + tobgp + alcgp, data = esoph)
model.extract(a, "response")
stopifnot(model.extract(a, "response") == model.response(a))

a <- model.frame(ncases/(ncases+ncontrols) ~ agegp + tobgp + alcgp,
                 data = esoph, weights = ncases+ncontrols)
model.response(a)
(mw <- model.extract(a, "weights"))
stopifnot(identical(unname(mw), model.weights(a)))

a <- model.frame(cbind(ncases,ncontrols) ~ agegp,
                 something = tobgp, data = esoph)
names(a)
stopifnot(model.extract(a, "something") == esoph$tobgp)
```

model.frame

model.frame`data.frame`formula...

model.frame(formula, ...)

Default S3 method:

```
model.frame(formula, data = NULL,
             subset = NULL, na.action,
             drop.unused.levels = FALSE, xlev = NULL, ...)
```

S3 method for class 'aovlist'

```
model.frame(formula, data = NULL, ...)
```

S3 method for class 'glm'

```
model.frame(formula, ...)
```

S3 method for class 'lm'

```
model.frame(formula, ...)
```

get_all_vars(formula, data, ...)

formula	<code>formulaterms</code>
data	<code>as.data.frame</code> formula
subset	<code>[.data.frame</code> dataformula
na.action	NA <code>na.action</code> data <code>na.action</code> options <code>na.fail</code> <code>na.omit</code> NULL
drop.unused.levels	FALSE
xlev	
...	model.frame <code>data</code> <code>na.action</code> subsetoffsetweights"(offset)" get_all_vars

formula"lm"model = TRUE"lqs"

formuladata"terms"formulaas.formulatermskeep.orderterms.formula

data

formulasubset...dataformula`formula()`subset`na.action`drop.unused.levelsxlevxlev

subset`polybs()``mean`

na.action = NULLNA

-

get_all_vars`data.frame`formula...model.frame.defaultformula

```
data.frameformula..."terms""terms"formula"na.action"NANa.pass
```

```
model.matrixformulamodel.extractexpand.model.frame
```

```
data.class(model.frame(dist ~ speed, data = cars))

## using a subset and an extra variable
model.frame(dist ~ speed, data = cars, subset = speed < 10, z = log(dist))

## get_all_vars(): new vars are recycled (iff length matches: 50 = 2*25)
ncars <- get_all_vars(sqrt(dist) ~ I(speed/2), data = cars, newVar = 2:3)
stopifnot(is.data.frame(ncars),
          identical(cars, ncars[,names(cars)]),
          ncol(ncars) == ncol(cars) + 1)
```

```
model.matrix
```

```
model.matrix
```

```
model.matrix(object, ...)

## Default S3 method:
model.matrix(object, data = environment(object),
             contrasts.arg = NULL, xlev = NULL, ...)
## S3 method for class 'lm'
model.matrix(object, ...)
```

```
object          terms
data            model.framemodel.frame
contrasts.arg   functioncontrastsdatafactor
xlev            model.frameadatamodel.frame
...
```

```
model.matrixterms(object)datamodel.frame(object)attr(terms(object),
"variables")data
contrasts.arg"contrasts"Ccontrastscontrasts.arg
~ a + b + b:aa
```



```
"assign"0term.labelstermsobject
"contrasts"
```

```
model.frame
model.extractterms
sparse.model.matrix
```

```
ff <- log(Volume) ~ log(Height) + log(Girth)
utils::str(m <- model.frame(ff, trees))
mat <- model.matrix(ff, m)

dd <- data.frame(a = gl(3,4), b = gl(4,1,12)) # balanced 2-way
options("contrasts") # typically 'treatment' (for unordered factors)
model.matrix(~ a + b, dd)
model.matrix(~ a + b, dd, contrasts.arg = list(a = "contr.sum"))
model.matrix(~ a + b, dd, contrasts.arg = list(a = "contr.sum", b = contr.poly))
m.orth <- model.matrix(~a+b, dd, contrasts.arg = list(a = "contr.helmert"))
crossprod(m.orth) # m.orth is ALMOST orthogonal
# invalid contrasts.. ignored with a warning:
stopifnot(identical(
  model.matrix(~ a + b, dd),
  model.matrix(~ a + b, dd, contrasts.arg = "contr.F00")))
```

model.tables	aov
--------------	-----

```
aov
```

```
model.tables(x, ...)

## S3 method for class 'aov'
model.tables(x, type = "effects", se = FALSE, cterms, ...)

## S3 method for class 'aovlist'
model.tables(x, type = "effects", se = FALSE, ...)
```

```
x          aov
type       "effects""means"
se
cterms
...
```

```
type = "effects"
type = "means"
"aov" "aovlist"
```

```
"tables.aov"
```

```
tables
```

```
n
```

```
se
```

```
aov
```

```
aovprojreplicationsTukeyHSDse.contrast
```

```
options(contrasts = c("contr.helmert", "contr.treatment"))
npk.aov <- aov(yield ~ block + N*P*K, npk)
model.tables(npk.aov, "means", se = TRUE)

## as a test, not particularly sensible statistically
npk.aovE <- aov(yield ~ N*P*K + Error(block), npk)
model.tables(npk.aovE, se = TRUE)
model.tables(npk.aovE, "means")
```

```
monthplot
```

```
monthplot(x, ...)
```

```
## S3 method for class 'stl'
monthplot(x, labels = NULL, ylab = choice, choice = "seasonal",
          ...)
```

```
## S3 method for class 'StructTS'
monthplot(x, labels = NULL, ylab = choice, choice = "sea", ...)
```

```
## S3 method for class 'ts'
monthplot(x, labels = NULL, times = time(x), phase = cycle(x),
          ylab = deparse1(substitute(x)), ...)
```

```
x
labels
ylab
times
phase
base
choice          stlStructTS
...
axes            add = TRUE
type            "h"
box             add = TRUE
add
colltylwd
col.baselyty.base
```

tsst1StructTS

```

require(graphics)

## The CO2 data
fit <- stl(log(co2), s.window = 20, t.window = 20)
plot(fit)
op <- par(mfrow = c(2,2))
monthplot(co2, ylab = "data", cex.axis = 0.8)
monthplot(fit, choice = "seasonal", cex.axis = 0.8)
monthplot(fit, choice = "trend", cex.axis = 0.8)
monthplot(fit, choice = "remainder", type = "h", cex.axis = 0.8)
par(op)

## The CO2 data, grouped quarterly
quarter <- (cycle(co2) - 1) %% 3
monthplot(co2, phase = quarter)

## see also JohnsonJohnson

```

mood.test

```

mood.test(x, ...)

## Default S3 method:
mood.test(x, y,
          alternative = c("two.sided", "less", "greater"), ...)

## S3 method for class 'formula'
mood.test(formula, data, subset, na.action, ...)

xy
alternative      "two.sided""greater""less"
formula          lhs ~ rhslhsrhs
data             model.frameformulaenvironment(formula)
subset
na.action        NAgetOption("na.action")
...

```

$$f(x-l)f((x-l)/s)/sls$$

$$s = 1$$

```

"htest"
statistic
p.value
alternative
method          "Mood two-sample test of scale"
data.name

```

[fligner.testansari.testvar.testbartlett.test](#)

```

## Same data as for the Ansari-Bradley test:
## Serum iron determination using Hyland control sera
ramsay <- c(111, 107, 100, 99, 102, 106, 109, 108, 104, 99,
            101, 96, 97, 102, 107, 113, 116, 113, 110, 98)
jung.parekh <- c(107, 108, 106, 98, 105, 103, 110, 105, 104,
                 100, 96, 108, 103, 104, 114, 114, 113, 108, 106, 99)
mood.test(ramsay, jung.parekh)
## Compare this to ansari.test(ramsay, jung.parekh)

```

Multinom

```

rmultinom(n, size, prob)
dmultinom(x, size = NULL, prob, log = FALSE)

```

```

x          K0:size
n
size       NKdmultinomsum(x)
prob       KK
log

```

```

xKdmultinom(x, prob)

```

$$P(X_1 = x_1, \dots, X_K = x_K) = C \times \prod_{j=1}^K \pi_j^{x_j}$$

```

CC = N!/(x1!...xK!)N = \sum_{j=1}^K x_j
X_jBin(size, prob[j])j = 1,...,K

```

```

rmultinom()X_jBin(n_j, P_j)n_1 = NsizeP_1 = \pi_1\pi\text{prob}j \geq 2n_j = N - \sum_{k=1}^{j-1} X_kP_j = \pi_j/(1 - \sum_{k=1}^{j-1} \pi_k)

```

`rmultinom()` $K \times nsize$

`dmultinom`

`dbinom`

```
rmultinom(10, size = 12, prob = c(0.1,0.2,0.8))
```

```
pr <- c(1,3,6,10) # normalization not necessary for generation
rmultinom(10, 20, prob = pr)
```

```
## all possible outcomes of Multinom(N = 3, K = 3)
X <- t(as.matrix(expand.grid(0:3, 0:3))); X <- X[, colSums(X) <= 3]
X <- rbind(X, 3:3 - colSums(X)); dimnames(X) <- list(letters[1:3], NULL)
X
round(apply(X, 2, function(x) dmultinom(x, prob = c(1,2,5))), 3)
```

`na.action`

`na.action(object, ...)`

`object` `NA`
`...`

`na.action``na.action.default``"na.action"``"na.action"`
`model.frame``NA``"na.action"`

`object``NA``NULL`

`options("na.action")``na.omit``na.fail``na.exclud``na.pass`

`na.action(na.omit(c(1, NA)))`

na.contiguous

na.contiguous(object, ...)

object

...

object

[na.omit](#)[na.omit.tsna.fail](#)

na.contiguous(presidents)

na.fail

[NA](#)[na.fail](#)[na.omit](#)[na.pass](#)

na.fail(object, ...)
na.omit(object, ...)
na.exclude(object, ...)
na.pass(object, ...)

object

...

na.omit"na.action""omit"

na.excludena.omit"na.action""exclude"[naresidnapredict](#)na.excludeNAa.exclude

[na.action](#)[options](#)[na.action](#)[lm](#)[glm](#)[na.contiguous](#)

```
DF <- data.frame(x = c(1, 2, 3), y = c(0, 10, NA))
na.omit(DF)
m <- as.matrix(DF)
na.omit(m)
stopifnot(all(na.omit(1:3) == 1:3)) # does not affect objects with no NA's
try(na.fail(DF)) #> Error: missing values in ...

options("na.action")
```

naprint

[na.action](#)

`naprint(x, ...)`

x [na.action](#)
...

`naprint.omit``naprint.default`

naresid

`naresid(omit, x, ...)`
`napredict(omit, x, ...)`

omit [na.action](#)"[na.action](#)"[na.omit](#)"[na.exclude](#)
x
...

[predictfittedresiduals](#)NA"lm""glm""nls"[factanalprcompprincomp](#)

na.excludeNA

naresidnapredictnaresidnapredict[weights](#)

x

na.omitna.exclude

NegBinomial

sizeprob

dnbinom(x, size, prob, mu, log = FALSE)
pnbinom(q, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
qnbinom(p, size, prob, mu, lower.tail = TRUE, log.p = FALSE)
rnbinom(n, size, prob, mu)

x

q

p

n length(n) > 1

size

prob 0 < prob <= 1

mu

loglog.p

lower.tail $P[X \leq x]P[X > x]$

size= nprob= p

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x$$

$x = 0, 1, 2, \dots, n > 0, 0 < p \leq 1$

$\mu = n(1-p)/p, n(1-p)/p^2$

[pgamma](#)(1 - prob)/probsizesize

musizeprobsize/(size+mu)mu + mu^2/size

xdnbinom

size == 0sizemuprobmu

$x F(x) \geq p F$

dnbinom
pnbinom
qnbinom
rnbinom

size
prob
NaN

nrnbinom

n

rnbinom
double

dnbinom
dbinom

pnbinom
pbeta

qnbinom

rnbinom

dbinom
dpois
dgeom

```
require(graphics)
```

```
x <- 0:11
```

```
dnbinom(x, size = 1, prob = 1/2) * 2^(1 + x) # == 1
```

```
126 / dnbinom(0:8, size = 2, prob = 1/2) #- theoretically integer
```

```
## Cumulative ('p') = Sum of discrete prob.s ('d'); Relative error :
```

```
summary(1 - cumsum(dnbinom(x, size = 2, prob = 1/2)) /
```

```
pnbinom(x, size = 2, prob = 1/2))
```

```
x <- 0:15
```

```
size <- (1:20)/4
```

```
persp(x, size, dnb <- outer(x, size, function(x,s) dnbinom(x, s, prob = 0.4)),
```

```
      xlab = "x", ylab = "s", zlab = "density", theta = 150)
```

```
title(tit <- "negative binomial density(x,s, pr = 0.4) vs. x & s")
```

```
image (x, size, log10(dnb), main = paste("log [", tit, "]"))
```

```
contour(x, size, log10(dnb), add = TRUE)
```

```
## Alternative parametrization
```

```
x1 <- rnbinom(500, mu = 4, size = 1)
```

```
x2 <- rnbinom(500, mu = 4, size = 10)
```

```
x3 <- rnbinom(500, mu = 4, size = 100)
```

```
h1 <- hist(x1, breaks = 20, plot = FALSE)
```

```
h2 <- hist(x2, breaks = h1$breaks, plot = FALSE)
```

```
h3 <- hist(x3, breaks = h1$breaks, plot = FALSE)
```

```
barplot(rbind(h1$counts, h2$counts, h3$counts),
```

```
      beside = TRUE, col = c("red", "blue", "cyan"),
```

```
      names.arg = round(h1$breaks[-length(h1$breaks)]))
```

nextn

nextnnfactors
nextn()[fft](#)factors

nextn(n, factors = c(2,3,5))

n "integer""double"
factors 2

[length](#)n"integer""double"

factorsnextn(91, c(2,6))nextn(91, c(2,3))
N <- nextn(..)2^53

[convolve](#)[fft](#)

nextn(1001) # 1024
table(nextn(599:630))
n <- 1:100 ; plot(n, nextn(n) - n, type = "o", lwd=2, cex=1/2)

nlm

f

nlm(f, p, ..., hessian = FALSE, typsize = rep(1, length(p)),
 fscale = 1, print.level = 0, ndigit = 12, gradtol = 1e-6,
 stepmax = max(1000 * sqrt(sum((p/typsize)^2)), 1000),
 steptol = 1e-6, iterlim = 100, check.analyticals = TRUE)

```

f          p...
          gradientgradienthessianderivgradienthessian
p
...        f
hessian    TRUEf
typsize
fscale     f
print.level 01
ndigit     f
gradtol     fp[i]p[i]
stepmax     stepmaxstepmax
steptol
iterlim
check.analyticals

```

```

...
check.analyticals = TRUE
choldc()

```

NANaN

```

minimum    f
estimate    f
gradient    f
hessian     f
code

```

estimateestimatesteptol

stepmaxstepmax

iterations

optimnlminb
constrOptimoptimizeunirootderiv
nls

```
f <- function(x) sum((x-1:length(x))^2)
nlm(f, c(10,10))
nlm(f, c(10,10), print.level = 2)
utils::str(nlm(f, c(5), hessian = TRUE))

f <- function(x, a) sum((x-a)^2)
nlm(f, c(10,10), a = c(3,5))
f <- function(x, a)
{
  res <- sum((x-a)^2)
  attr(res, "gradient") <- 2*(x-a)
  res
}
nlm(f, c(10,10), a = c(3,5))

## more examples, including the use of derivatives.
## Not run: demo(nlm)
```

nlminb

```
nlminb(start, objective, gradient = NULL, hessian = NULL, ...,
        scale = 1, control = list(), lower = -Inf, upper = Inf)
```

```
start
objective      objectivestartobjective...
gradient       objectiveobjectivestart
hessian        objectiveobjectivelength(start)
...            objective
scale
control
lowerupper     start
```

```
startobjectivegradienthessian
NaNNaN+Inf
```

```
par
objective      objectivepar
convergence    0
message        NULL
iterations
evaluations
```

```
control

eval.max
iter.max
trace
abs.tol 1e-20
rel.tol 1e-10
x.tol 1.5e-8
xf.tol 2.2e-14
step.min, step.max 1.
rel.tol
```

<https://netlib.org/port/>

[optimnlm](#)
[optimizeconstrOptim](#)

```

x <- rnbino(100, mu = 10, size = 10)
hdev <- function(par)
  -sum(dnbinom(x, mu = par[1], size = par[2], log = TRUE))
nlminb(c(9, 12), hdev)
nlminb(c(20, 20), hdev, lower = 0, upper = Inf)
nlminb(c(20, 20), hdev, lower = 0.001, upper = Inf)

## slightly modified from the S-PLUS help page for nlminb
# this example minimizes a sum of squares with known solution y
sumsq <- function( x, y) {sum((x-y)^2)}
y <- rep(1,5)
x0 <- rnorm(length(y))
nlminb(start = x0, sumsq, y = y)
# now use bounds with a y that has some components outside the bounds
y <- c( 0, 2, 0, -2, 0)
nlminb(start = x0, sumsq, lower = -1, upper = 1, y = y)
# try using the gradient
sumsq.g <- function(x, y) 2*(x-y)
nlminb(start = x0, sumsq, sumsq.g,
  lower = -1, upper = 1, y = y)
# now use the hessian, too
sumsq.h <- function(x, y) diag(2, nrow = length(x))
nlminb(start = x0, sumsq, sumsq.g, sumsq.h,
  lower = -1, upper = 1, y = y)

## Rest lifted from optim help page

fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
grr <- function(x) { ## Gradient of 'fr'
  x1 <- x[1]
  x2 <- x[2]
  c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
    200 * (x2 - x1 * x1))
}
nlminb(c(-1.2,1), fr)
nlminb(c(-1.2,1), fr, grr)

flb <- function(x)
  { p <- length(x); sum(c(1, rep(4, p-1)) * (x - c(1, x[-p]))^2)^2 }
## 25-dimensional box constrained
## par[24] is *not* at boundary
nlminb(rep(3, 25), flb, lower = rep(2, 25), upper = rep(4, 25))
## trying to use a too small tolerance:
r <- nlminb(rep(3, 25), flb, control = list(rel.tol = 1e-16))
stopifnot(grepl("rel.tol", r$message))

```

```
nls(formula, data, start, control, algorithm,
    trace, subset, weights, na.action, model,
    lower, upper, ...)
```

```
formula
data          formulaweights
start          startformula$selfStartstartalgorithm != "plinear"
control        listnls.control
algorithm      "plinear""port"
trace          FALSETRUEformat()getOption("digits")"plinear""port"
subset
weights
na.action      NAna.actionoptionsna.failna.omitna.exclude
model          FALSE
lowerupper     start"port"
...
```

```
nlsanovacoefconfintdeviancedf.residualfittedformulaalogLikpredictprintprofile
residualsummaryvcovweights
formulaweightsdataformulaformulaformula
subsetna.actiondata
anova
```

```
m          nlsModel
data        nlsenvironmentmenvironment(m$conv)
call        algorithm
na.action   "na.action"
dataClasses "dataClasses""terms"
model       model = TRUE
weights     weights
convInfo
control     listcontrol
convergence message
            algorithm = "port"0
            convInfo
```



```
nls
```

```
nls
```

$$y = f(x, \theta) + \varepsilon$$

```
var( $\varepsilon$ ) > 0
```

$$y = f(x, \theta)$$

```
scaleOffsetcontrolalgorithm = "port"
```

```
algorithm = "port"
```

```
warnOnly = TRUEcontrolnls.control
```

```
algorithm = "port"
```

```
https://netlib.org/port/
```

```
summary.nlspredict.nlsprofile.nls
```

```
selfStart
```

```
require(graphics)
```

```
DNase1 <- subset(DNase, Run == 1)
```

```
## using a selfStart model
```

```
fm1DNase1 <- nls(density ~ SSlogis(log(conc), Asym, xmid, scal), DNase1)
```

```
summary(fm1DNase1)
```

```
## the coefficients only:
```

```
coef(fm1DNase1)
```

```
## including their SE, etc:
```

```
coef(summary(fm1DNase1))
```

```
## using conditional linearity
```

```
fm2DNase1 <- nls(density ~ 1/(1 + exp((xmid - log(conc))/scal)),
```

```
data = DNase1,
```

```
start = list(xmid = 0, scal = 1),
```

```
algorithm = "plinear")
```

```
summary(fm2DNase1)
```

```
## without conditional linearity
```

```
fm3DNase1 <- nls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
```

```
data = DNase1,
```

```
start = list(Asym = 3, xmid = 0, scal = 1))
```

```
summary(fm3DNase1)
```

```

## using Port's nl2sol algorithm
fm4DNase1 <- nls(density ~ Asym/(1 + exp((xmid - log(conc))/scal)),
                data = DNase1,
                start = list(Asym = 3, xmid = 0, scal = 1),
                algorithm = "port")
summary(fm4DNase1)

## weighted nonlinear regression
Treated <- Puromycin[Puromycin$state == "treated", ]
weighted.MM <- function(resp, conc, Vm, K)
{
  ## Purpose: exactly as white book p. 451 -- RHS for nls()
  ## Weighted version of Michaelis-Menten model
  ## -----
  ## Arguments: 'y', 'x' and the two parameters (see book)
  ## -----
  ## Author: Martin Maechler, Date: 23 Mar 2001

  pred <- (Vm * conc)/(K + conc)
  (resp - pred) / sqrt(pred)
}

Pur.wt <- nls( ~ weighted.MM(rate, conc, Vm, K), data = Treated,
              start = list(Vm = 200, K = 0.1))
summary(Pur.wt)

## Passing arguments using a list that can not be coerced to a data.frame
lisTreat <- with(Treated,
                 list(conc1 = conc[1], conc.1 = conc[-1], rate = rate))

weighted.MM1 <- function(resp, conc1, conc.1, Vm, K)
{
  conc <- c(conc1, conc.1)
  pred <- (Vm * conc)/(K + conc)
  (resp - pred) / sqrt(pred)
}
Pur.wt1 <- nls( ~ weighted.MM1(rate, conc1, conc.1, Vm, K),
              data = lisTreat, start = list(Vm = 200, K = 0.1))
stopifnot(all.equal(coef(Pur.wt), coef(Pur.wt1)))

## Chambers and Hastie (1992) Statistical Models in S (p. 537):
## If the value of the right side [of formula] has an attribute called
## 'gradient' this should be a matrix with the number of rows equal
## to the length of the response and one column for each parameter.

weighted.MM.grad <- function(resp, conc1, conc.1, Vm, K)
{
  conc <- c(conc1, conc.1)

  K.conc <- K+conc
  dy.dV <- conc/K.conc
  dy.dK <- -Vm*dy.dV/K.conc
  pred <- Vm*dy.dV
  pred.5 <- sqrt(pred)
  dev <- (resp - pred) / pred.5
  Ddev <- -0.5*(resp+pred)/(pred.5*pred)
  attr(dev, "gradient") <- Ddev * cbind(Vm = dy.dV, K = dy.dK)
}

```

```

    dev
  }

Pur.wt.grad <- nls( ~ weighted.MM.grad(rate, conc1, conc.1, Vm, K),
                  data = lisTreat, start = list(Vm = 200, K = 0.1))

rbind(coef(Pur.wt), coef(Pur.wt1), coef(Pur.wt.grad))

## In this example, there seems no advantage to providing the gradient.
## In other cases, there might be.

## The two examples below show that you can fit a model to
## artificial data with noise but not to artificial data
## without noise.
x <- 1:10
y <- 2*x + 3                                # perfect fit
## terminates in an error, because convergence cannot be confirmed:
try(nls(y ~ a + b*x, start = list(a = 0.12345, b = 0.54321)))
## adjusting the convergence test by adding 'scaleOffset' to its denominator RSS:
nls(y ~ a + b*x, start = list(a = 0.12345, b = 0.54321),
    control = list(scaleOffset = 1, printEval=TRUE))
## Alternatively jittering the "too exact" values, slightly:
set.seed(27)
yeps <- y + rnorm(length(y), sd = 0.01) # added noise
nls(yeps ~ a + b*x, start = list(a = 0.12345, b = 0.54321))

## the nls() internal cheap guess for starting values can be sufficient:
x <- -(1:100)/10
y <- 100 + 10 * exp(x / 2) + rnorm(x)/10
nlmod <- nls(y ~ Const + A * exp(B * x))

plot(x,y, main = "nls(*), data, true function and fit, n=100")
curve(100 + 10 * exp(x / 2), col = 4, add = TRUE)
lines(x, predict(nlmod), col = 2)

## Here, requiring close convergence, must use more accurate numerical differentiation,
## as this typically gives Error: "step factor .. reduced below 'minFactor' .."

try(nlm1 <- update(nlmod, control = list(tol = 1e-7)))
o2 <- options(digits = 10) # more accuracy for 'trace'
## central differencing works here typically (PR#18165: not converging on *some*):
ctr2 <- nls.control(nDcentral=TRUE, tol = 8e-8, # <- even smaller than above
  warnOnly =
    TRUE || # << work around; e.g. needed on some ATLAS-Lapack setups
    (grepl("^aarch64.*linux", R.version$platform) && grepl("^NixOS", osVersion)
    ))
(nlm2 <- update(nlmod, control = ctr2, trace = TRUE)); options(o2)
## --> convergence tolerance 4.997e-8 (in 11 iter.)

## The muscle dataset in MASS is from an experiment on muscle
## contraction on 21 animals. The observed variables are Strip
## (identifier of muscle), Conc (Cacl concentration) and Length
## (resulting length of muscle section).
```

```

if(requireNamespace("MASS", quietly = TRUE)) withAutoprint({
## The non linear model considered is
##      Length = alpha + beta*exp(-Conc/theta) + error
## where theta is constant but alpha and beta may vary with Strip.

with(MASS::muscle, table(Strip)) # 2, 3 or 4 obs per strip

## We first use the plinear algorithm to fit an overall model,
## ignoring that alpha and beta might vary with Strip.
musc.1 <- nls(Length ~ cbind(1, exp(-Conc/th)), MASS::muscle,
              start = list(th = 1), algorithm = "plinear")
summary(musc.1)

## Then we use nls' indexing feature for parameters in non-linear
## models to use the conventional algorithm to fit a model in which
## alpha and beta vary with Strip. The starting values are provided
## by the previously fitted model.
## Note that with indexed parameters, the starting values must be
## given in a list (with names):
b <- coef(musc.1)
musc.2 <- nls(Length ~ a[Strip] + b[Strip]*exp(-Conc/th), MASS::muscle,
              start = list(a = rep(b[2], 21), b = rep(b[3], 21), th = b[1]))
summary(musc.2)
})

```

nls.control	nls
-------------	-----

[nls](#)

```

nls.control(maxiter = 50, tol = 1e-05, minFactor = 1/1024,
            printEval = FALSE, warnOnly = FALSE, scaleOffset = 0,
            nDcentral = FALSE)

```

maxiter

tol

minFactor

printEval

warnOnly [nls\(\)](#)maxiterminFactor

scaleOffset 0nls()1

nDcentral [logicalnumericDeriv](#)(*, central=TRUE)

NLSstClosestX

xyyval

NLSstClosestX(xy, yval)

xy sortedXyData

yval y

x

[sortedXyDataNLSstLfAsymptoteNLSstRtAsymptoteselfStart](#)

```
DNase.2 <- DNase[ DNase$Run == "2", ]  
DN.srt <- sortedXyData(expression(log(conc)), expression(density), DNase.2)  
NLSstClosestX(DN.srt, 1.0)
```

NLSstLfAsymptote

xyxyinitial

NLSstLfAsymptote(xy)

xy sortedXyData

x

[sortedXyDataNLSstClosestXNLSstRtAsymptoteselfStart](#)

```
DNase.2 <- DNase[ DNase$Run == "2", ]  
DN.srt <- sortedXyData( expression(log(conc)), expression(density), DNase.2 )  
NLSstLfAsymptote( DN.srt )
```

NLSstRtAsymptote

xyxyinitial

NLSstRtAsymptote(xy)

xy sortedXyData

x

[sortedXyDataNLSstClosestXNLSstLfAsymptoteselfStart](#)

```
DNase.2 <- DNase[ DNase$Run == "2", ]
DN.srt <- sortedXyData( expression(log(conc)), expression(density), DNase.2 )
NLSstRtAsymptote( DN.srt )
```

nobs

[AIC](#)

nobs(object, ...)

```
## Default S3 method:
nobs(object, use.fallback = FALSE, ...)
```

object

use.fallback

...

```
"lm" "glm" "nls" "logLik" use.fallback = TRUE weights residuals
nobs stepAIC drop1
lm glm nls
```

NA

[AIC](#)

Normal

meansd

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

xq

p

n length(n) > 1

mean

sd

loglog.p

lower.tail $P[X \leq x]P[X > x]$

meansd01

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

$\mu\sigma$

dnormpnormqnormrnorm

nrnorm

n

sd = 0sdmud < 0NaN

pnorm

qnorm

log.p=FALSE

<https://CRAN.R-project.org/package=DPQ/vignettes/qnorm-asymp.pdf>

rnorm

dlnorm

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

numericDeriv

numericDeriv

```
numericDeriv(expr, theta, rho = parent.frame(), dir = 1,
             eps = .Machine$double.eps ^ (1/if(central) 3 else 2), central = FALSE)
```

expr	expressioncallnumeric
theta	characterexpr
rho	environmentexpr
dir	-1, 1theta
eps	$h(f(x+h) - f(x))/h$ central
central	$(f(x+h) - f(x-h))/2hf()$

```
numeric_deriv
doubleinteger
```

```
eval(expr, envir = rho)"gradient"theta
```

```
<saikat@stat.wisc.edu>epscentral
```

```
myenv <- new.env()
myenv$mean <- 0.
myenv$sd <- 1.
myenv$x <- seq(-3., 3., length.out = 31)
nD <- numericDeriv(quote(pnorm(x, mean, sd)), c("mean", "sd"), myenv)
str(nD)

## Visualize :
require(graphics)
matplot(myenv$x, cbind(c(nD), attr(nD, "gradient")), type="l")
abline(h=0, lty=3)
## "gradient" is close to the true derivatives, you don't see any diff.:
curve( - dnorm(x), col=2, lty=3, lwd=2, add=TRUE)
curve(-x*dnorm(x), col=3, lty=3, lwd=2, add=TRUE)
##
# shows 1.609e-8 on most platforms
all.equal(attr(nD,"gradient"),
          with(myenv, cbind(-dnorm(x), -x*dnorm(x))))
```

offset

```
offset(object)
```

```
object
```

```
-offset+
```

```
model.offsetmodel.frame
glmInsurance
```

`oneway.test`

```
oneway.test(formula, data, subset, na.action, var.equal = FALSE)
```

<code>formula</code>	<code>lhs ~ rhs</code>
<code>data</code>	<code>model.frame</code>
<code>subset</code>	<code>formulaenvironment(formula)</code>
<code>na.action</code>	<code>NAgetOption("na.action")</code>
<code>var.equal</code>	<code>TRUE</code>

`"htest"`

`statistic`

`parameter`

`p.value`

`method`

`data.name`

`t.test`

```
## Not assuming equal variances
oneway.test(extra ~ group, data = sleep)
## Assuming equal variances
oneway.test(extra ~ group, data = sleep, var.equal = TRUE)
## which gives the same result as
anova(lm(extra ~ group, data = sleep))
```

optim

```
optim(par, fn, gr = NULL, ...,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN",
                  "Brent"),
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)
```

```
optimHess(par, fn, gr = NULL, ..., control = list())
```

```
par
fn
gr          "BFGS" "CG" "L-BFGS-B" NULL
           "SANN" NULL
...         fngr
method
lowerupper "L-BFGS-B" "Brent"
control    list
hessian
```

```
...
optimcontrol$fnscaleoptimHesshessian = TRUE
```

```
"BFGS"
"CG"
"L-BFGS-B"
```

```
"SANN" "SANN" temp / log(((t-1) %% tmax)*tmax + exp(1)) ttemptmax control "SANN"
"Brent" optimize(<ff>, lower, upper, tol = control$reltol) <ff> function(par)
fn(par, ...)/control$fnscaleoptim() method mle
fnNAInfn "L-BFGS-B"
optimpar
control
trace "L-BFGS-B"
fnscale fngrfn(par)/fnscale
parscale par/parscale method = "Brent"
```

```

ndeps par/parscale1e-3
maxit 100500"Nelder-Mead"
      "SANN"maxit10000

abstol

reltol reltol * (abs(val) + reltol)sqrt(.Machine$double.eps)1e-8
alphabeta gamma "Nelder-Mead"alphabeta gamma
REPORT "BFGS""L-BFGS-B""SANN"control$trace"BFGS""L-BFGS-B""SANN"
warn.1d.NelderMead logical"Nelder-Mead"
type 123
lmm "L-BFGS-B"5
factr "L-BFGS-B"1e71e-8
pgtol "L-BFGS-B"
temp "SANN"10
tmax "SANN"10

parfng rpar
fn

optim

par
value      fnpar
counts     fngrfn
convergence 0"SANN""Brent"
           1 maxit
           10
           51 "L-BFGS-B"message
           52 "L-BFGS-B"message
message    NULL
hessian    hessian

optimHesshessian

optimpar"Brent"optimize"BFGS"

"Nelder-Mead""BFGS""CG"p2c
"L-BFGS-B"opt/lbfgs_bcm.shartoms/778
"SANN"

```

R^d

nlmnlminb

optimizeconstrOptim

```
require(graphics)

fr <- function(x) { ## Rosenbrock Banana function
  x1 <- x[1]
  x2 <- x[2]
  100 * (x2 - x1 * x1)^2 + (1 - x1)^2
}
grr <- function(x) { ## Gradient of 'fr'
  x1 <- x[1]
  x2 <- x[2]
  c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
    200 * (x2 - x1 * x1))
}
optim(c(-1.2,1), fr)
(res <- optim(c(-1.2,1), fr, grr, method = "BFGS"))
optimHess(res$par, fr, grr)
optim(c(-1.2,1), fr, NULL, method = "BFGS", hessian = TRUE)
## These do not converge in the default number of steps
optim(c(-1.2,1), fr, grr, method = "CG")
optim(c(-1.2,1), fr, grr, method = "CG", control = list(type = 2))
optim(c(-1.2,1), fr, grr, method = "L-BFGS-B")

flb <- function(x)
  { p <- length(x); sum(c(1, rep(4, p-1)) * (x - c(1, x[-p])^2)^2) }
## 25-dimensional box constrained
optim(rep(3, 25), flb, NULL, method = "L-BFGS-B",
      lower = rep(2, 25), upper = rep(4, 25)) # par[24] is *not* at boundary

## "wild" function , global minimum at about -15.81515
fw <- function (x)
  10*sin(0.3*x)*sin(1.3*x^2) + 0.00001*x^4 + 0.2*x+80
plot(fw, -50, 50, n = 1000, main = "optim() minimising 'wild function'")

res <- optim(50, fw, method = "SANN",
            control = list(maxit = 20000, temp = 20, parscale = 20))
res
## Now improve locally {typically only by a small bit}:
(r2 <- optim(res$par, fw, method = "BFGS"))
points(r2$par, r2$value, pch = 8, col = "red", cex = 2)
```

```

## Combinatorial optimization: Traveling salesman problem
library(stats) # normally loaded

eurodistmat <- as.matrix(eurodist)

distance <- function(sq) { # Target function
  sq2 <- embed(sq, 2)
  sum(eurodistmat[cbind(sq2[,2], sq2[,1])])
}

genseq <- function(sq) { # Generate new candidate sequence
  idx <- seq(2, NROW(eurodistmat)-1)
  changepoints <- sample(idx, size = 2, replace = FALSE)
  tmp <- sq[changepoints[1]]
  sq[changepoints[1]] <- sq[changepoints[2]]
  sq[changepoints[2]] <- tmp
  sq
}

sq <- c(1:nrow(eurodistmat), 1) # Initial sequence: alphabetic
distance(sq)
# rotate for conventional orientation
loc <- -cmdscale(eurodist, add = TRUE)$points
x <- loc[,1]; y <- loc[,2]
s <- seq_len(nrow(eurodistmat))
tspinit <- loc[sq,]

plot(x, y, type = "n", asp = 1, xlab = "", ylab = "",
     main = "initial solution of traveling salesman problem", axes = FALSE)
arrows(tspinit[s,1], tspinit[s,2], tspinit[s+1,1], tspinit[s+1,2],
      angle = 10, col = "green")
text(x, y, labels(eurodist), cex = 0.8)

set.seed(123) # chosen to get a good soln relatively quickly
res <- optim(sq, distance, genseq, method = "SANN",
            control = list(maxit = 30000, temp = 2000, trace = TRUE,
                          REPORT = 500))
res # Near optimum distance around 12842

tspres <- loc[res$par,]
plot(x, y, type = "n", asp = 1, xlab = "", ylab = "",
     main = "optim() 'solving' traveling salesman problem", axes = FALSE)
arrows(tspres[s,1], tspres[s,2], tspres[s+1,1], tspres[s+1,2],
      angle = 10, col = "red")
text(x, y, labels(eurodist), cex = 0.8)

## 1-D minimization: "Brent" or optimize() being preferred.. but NM may be ok and "unavoidable",
## ----- so we can suppress the check+warning :
system.time(ro <- optimize(function(x) (x-pi)^2, c(0, 10)))
system.time(ro <- optim(1, function(x) (x-pi)^2, control=list(warn.1d.NelderMead = FALSE)))
ro$minimum - pi # 0 (perfect), on one platform
ro$par - pi # ~ 1.9e-4 on one platform
utils::str(ro)

```

optimize

optimizelowerupperf

optimiseoptimize

```
optimize(f, interval, ..., lower = min(interval), upper = max(interval),
        maximum = FALSE,
        tol = .Machine$double.eps^0.25)
```

```
optimise(f, interval, ..., lower = min(interval), upper = max(interval),
        maximum = FALSE,
        tol = .Machine$double.eps^0.25)
```

f maximum

interval

... f

lower

upper

maximum

tol

...

flowerupper

$f\epsilon|x_0| + (tol/3)\epsilon\sqrt{.Machine\$double.eps}x_0$ optimize()\$minimum

$ff\epsilon|x| + (tol/3)x_0$ flower,upper $\epsilon|x_0| + tol$

foptimize()

$fx_1 = a + (1 - \phi)(b - a)$ (a,b) = (lower, upper) $\phi = (\sqrt{5} - 1)/2 = 0.61803..x_2 = a + \phi(b - a)$

$[x_1, x_2]$ f

ff(, ...)

f

minimummaximumobjective

<https://netlib.org/fmm/fmin.f>localmin

nlmuniroot


```

require(graphics)

f <- function (x, a) (x - a)^2
xmin <- optimize(f, c(0, 1), tol = 0.0001, a = 1/3)
xmin

## See where the function is evaluated:
optimize(function(x) x^2*(print(x)-1), lower = 0, upper = 10)

## "wrong" solution with unlucky interval and piecewise constant f():
f <- function(x) ifelse(x > -1, ifelse(x < 4, exp(-1/abs(x - 1)), 10), 10)
fp <- function(x) { print(x); f(x) }

plot(f, -2,5, ylim = 0:1, col = 2)
optimize(fp, c(-4, 20)) # doesn't see the minimum
optimize(fp, c(-7, 20)) # ok

```

order.dendrogram

"label"

order.dendrogram(x)

S3 method for class 'dendrogram'
labels(object, ...)

xobject [as.dendrogram](#)

...

r <- order.dendrogram()

order.dendrogramlabels.dendrogram

[reorderdendrogram](#)

```

set.seed(123)
x <- rnorm(10)
hc <- hclust(dist(x))
hc$order
dd <- as.dendrogram(hc)
order.dendrogram(dd) ## the same :
stopifnot(hc$order == order.dendrogram(dd))

d2 <- as.dendrogram(hclust(dist(USArrests)))
labels(d2) ## in this case the same as
stopifnot(identical(labels(d2),
  rownames(USArrests)[order.dendrogram(d2)]))

```

p.adjust

```
p.adjust(p, method = p.adjust.methods, n = length(p))
```

```

p.adjust.methods
# c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY",
#   "fdr", "none")

```

```

p           NAas.numeric
method      character
n           length(p)

```

```
"bonferroni""holm""hochberg""hommel""BH""fdr""BY""none"p.adjust.methodsp.adjust
```

```
"BH""fdr""BY"
```

```
"BY" $qq/\sum_{i=1}^m 1/im$ "BH"p"BH"
```

```
nlength(p)"bonferroni""holm"
```

```
pp
```

<https://www.jstor.org/stable/4615733>

pairwise.*pairwise.t.test

```
require(graphics)

set.seed(123)
x <- rnorm(50, mean = c(rep(0, 25), rep(3, 25)))
p <- 2*pnorm(sort(-abs(x)))

round(p, 3)
round(p.adjust(p), 3)
round(p.adjust(p, "BH"), 3)

## or all of them at once (dropping the "fdr" alias):
p.adjust.M <- p.adjust.methods[p.adjust.methods != "fdr"]
p.adj <- sapply(p.adjust.M, function(meth) p.adjust(p, meth))
p.adj.60 <- sapply(p.adjust.M, function(meth) p.adjust(p, meth, n = 60))
stopifnot(identical(p.adj[, "none"], p), p.adj <= p.adj.60)

round(p.adj, 3)
## or a bit nicer:
head(round(100 * p.adj[, c(7, 1:6)], 2), n=21) # in [percent]:
##      none holm hochberg hommel bonferroni BH BY
## [1,] 0.00 0.00      0.00  0.00      0.00 0.00 0.01 *)
## [2,] 0.00 0.10      0.10  0.10      0.11 0.04 0.19 *)
## [3,] 0.00 0.12      0.12  0.12      0.13 0.04 0.19 *)
## [4,] 0.01 0.46      0.46  0.42      0.49 0.09 0.43
## [5,] 0.01 0.48      0.48  0.45      0.53 0.09 0.43
## ....
## [18,] 0.88 29.06     29.06 27.30     44.03 2.45 11.00
## [19,] 0.94 30.08     30.08 29.14     47.01 2.47 11.13
## [20,] 1.13 35.02     35.02 33.89     56.49 2.82 12.71
## [21,] 2.12 63.45     63.45 57.11    100.00 5.04 22.66
##
## *) The smallest 3 Bonferroni values are smaller than the "BY" ones,
##    (John Maindonald, PR#17136)

## number of rejected H0 ("P" < 0.05):
colSums(p.adj < 0.05)
```

```

## holm    hochberg    hommel bonferroni    BH    BY    none
##    11         11         11         11    20    12    22

## visual comparison
matplot(p, p.adj, ylab="p.adjust(p, meth)", type = "l", asp = 1, lty = 1:6,
        col = 1:7, main = "P-value adjustments")
legR <- function() {
  legend("bottomright", p.adjust.M, col = 1:7, lty = 1:6, bty = "n", inset = 0.05)
  rug(p) }
legR()

## zoom in & log scale
lim <- c(7e-7, .20)
matplot(p, p.adj, ylab="p.adjust(p, meth)", type = "l", asp = 1, lty = 1:6, col = 1:7,
        main = "P-value adjustments [log-log]", log = "xy", xlim=lim, ylim=lim, las=1)
legR()

## Can work with NAs:
pN <- p; iN <- c(46, 47); pN[iN] <- NA
pN.a <- sapply(p.adjust.M, function(meth) p.adjust(pN, meth))
## The smallest 20 P-values all affected by the NA's :
round((pN.a / p.adj)[1:20, ], 4)

```

Pair

"Pair"

Pair(x, y)

x

y x

"Pair"

[t.testwilcox.test](#)

`pairwise.prop.test`

```
pairwise.prop.test(x, n, p.adjust.method = p.adjust.methods, ...)
```

```
x
n      x
p.adjust.method
      p.adjust
...      prop.test
```

```
"pairwise.htest"
```

```
prop.testp.adjust
```

```
smokers <- c( 83, 90, 129, 70 )
patients <- c( 86, 93, 136, 82 )
pairwise.prop.test(smokers, patients)
```

`pairwise.t.test`

```
pairwise.t.test(x, g, p.adjust.method = p.adjust.methods,
               pool.sd = !paired, paired = FALSE,
               alternative = c("two.sided", "less", "greater"),
               ...)
```

```
x
g
p.adjust.method
      p.adjust
pool.sd
paired
alternative    "two.sided" "greater" "less"
...           t.test
```

```
pool.sdt.testpool.sdpairedTRUE  
alternative"two.sided"g
```

```
"pairwise.htest"
```

```
t.testp.adjust
```

```
attach(airquality)  
Month <- factor(Month, labels = month.abb[5:9])  
pairwise.t.test(Ozone, Month)  
pairwise.t.test(Ozone, Month, p.adjust.method = "bonf")  
pairwise.t.test(Ozone, Month, pool.sd = FALSE)  
detach()
```

```
pairwise.table
```

```
pairwise.table(compare.levels, level.names, p.adjust.method)
```

```
compare.levels functionij  
level.names  
p.adjust.method  
p.adjust.methods
```

```
compare.levelsij
```

```
pairwise.t.test
```

`pairwise.wilcox.test`

```
pairwise.wilcox.test(x, g, p.adjust.method = p.adjust.methods,
                     paired = FALSE, ...)
```

```
x
g
p.adjust.method
p.adjust
paired
...
wilcox.test
```

```
wilcox.testalternative="two.sided" g
```

```
"pairwise.htest"
```

```
wilcox.testp.adjust
```

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
## These give warnings because of ties :
pairwise.wilcox.test(Ozone, Month)
pairwise.wilcox.test(Ozone, Month, p.adjust.method = "bonf")
detach()
```

`plot.acf`

```
"acf"
```

```
## S3 method for class 'acf'
plot(x, ci = 0.95, type = "h", xlab = "Lag", ylab = NULL,
     ylim = NULL, main = NULL,
     ci.col = "blue", ci.type = c("white", "ma"),
     max.mfrow = 6, ask = Npgs > 1 && dev.interactive(),
     mar = if(nser > 2) c(3,2,2,0.8) else par("mar"),
     oma = if(nser > 2) c(1,1.2,1,1) else par("oma"),
     mgp = if(nser > 2) c(1.5,0.6,0) else par("mgp"),
     xpd = par("xpd"),
     cex.main = if(nser > 2) 1 else par("cex.main"),
     verbose = getOption("verbose"),
     ...)
```

```

x          "acf"
ci         ci
type
xlab
ylab
ylim
main
ci.col
ci.type     $kk - 1$ 
max.mfrow  xpar(mfrow = c(m,m))
ask        TRUE
maromamgpxpdex.main
           par(*)x
verbose
...

```

```
plot.acfci.type = "ma"
```

```
acfplot.acf
```

```

require(graphics)

z4 <- ts(matrix(rnorm(400), 100, 4), start = c(1961, 1), frequency = 12)
z7 <- ts(matrix(rnorm(700), 100, 7), start = c(1961, 1), frequency = 12)
acf(z4)
acf(z7, max.mfrow = 7) # squeeze onto 1 page
acf(z7) # multi-page

```

```
plot.density
```

```
plot
```

```

## S3 method for class 'density'
plot(x, main = NULL, xlab = NULL, ylab = "Density", type = "l",
     zero.line = TRUE, ...)

```

```

x          "density"
mainxlabylabtype

```

```

...
zero.line  TRUE  $y = 0$ 

```


density

plot.HoltWinters	"HoltWinters"
------------------	---------------

```
## S3 method for class 'HoltWinters'
plot(x, predicted.values = NA, intervals = TRUE,
      separator = TRUE, col = 1, col.predicted = 2,
      col.intervals = 4, col.separator = 1, lty = 1,
      lty.predicted = 1, lty.intervals = 1, lty.separator = 3,
      ylab = "Observed / Fitted",
      main = "Holt-Winters filtering",
      ylim = NULL, ...)
```

```
x                "HoltWinters"
predicted.values
                  predict.HoltWinters
intervals        TRUE
separator        TRUE
collty
col.predictedlty.predicted

col.intervalslty.intervals

col.separatorlty.separator

ylab
main
ylim        NULL
...
```

<David.Meyer@wu.ac.at>

HoltWinterspredict.HoltWinters

plot.isoreg

isoreg

[plotlinesisoreg](#)

```
## S3 method for class 'isoreg'
plot(x, plot.type = c("single", "row.wise", "col.wise"),
     main = paste("Isotonic regression", deparse(x$call)),
     main2 = "Cumulative Data and Convex Minorant",
     xlab = "x0", ylab = "x$y",
     par.fit = list(col = "red", cex = 1.5, pch = 13, lwd = 1.5),
     mar = if (both) 0.1 + c(3.5, 2.5, 1, 1) else par("mar"),
     mgp = if (both) c(1.6, 0.7, 0) else par("mgp"),
     grid = length(x$x) < 12, ...)
```

```
## S3 method for class 'isoreg'
lines(x, col = "red", lwd = 1.5,
      do.points = FALSE, cex = 1.5, pch = 13, ...)
```

x	isoreg
plot.type	
main	title
main2	
xlabylab	
par.fit	listpointslines
mar	par
mgp	
grid	grid()
do.points	lines() plot()
collwdcexpch	lines() cexpch do.points TRUE
...	

[isoregisoreg](#)

```
require(graphics)
```

```
utils::example(isoreg) # for the examples there
```

```
plot(y3, main = "simple plot(.) + lines(<isoreg>")
lines(ir3)
```

```
## 'same' plot as above, "proving" that only ranks of 'x' are important
plot(isoreg(2^(1:9), c(1,0,4,3,3,5,4,2,0)), plot.type = "row", log = "x")
```

```

plot(ir3, plot.type = "row", ylab = "y3")
plot(isoreg(y3 - 4), plot.type = "r", ylab = "y3 - 4")
plot(ir4, plot.type = "ro", ylab = "y4", xlab = "x = 1:n")

## experiment a bit with these (C-c C-j):
plot(isoreg(sample(9), y3), plot.type = "row")
plot(isoreg(sample(9), y3), plot.type = "col.wise")

plot(ir <- isoreg(sample(10), sample(10, replace = TRUE)),
      plot.type = "r")

```

plot.lm	lm
---------	----

which $\sqrt{|residuals|}$ 5

```

## S3 method for class 'lm'
plot(x, which = c(1,2,3,5),
      caption = list("Residuals vs Fitted", "Q-Q Residuals",
                     "Scale-Location", "Cook's distance",
                     "Residuals vs Leverage",
                     expression("Cook's dist vs Leverage* " * h[ii] / (1 - h[ii]))),
      panel = if(add.smooth) function(x, y, ...)
        panel.smooth(x, y, iter=iter.smooth, ...) else points,
      sub.caption = NULL, main = "",
      ask = prod(par("mfcol")) < length(which) && dev.interactive(),
      ...,
      id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75,
      qqline = TRUE, cook.levels = c(0.5, 1.0),
      cook.col = 8, cook.lty = 2, cook.legendChanges = list(),
      add.smooth = getOption("add.smooth"),
      iter.smooth = if(isGlm) 0 else 3,
      label.pos = c(4,2),
      cex.caption = 1, cex.oma.main = 1.25
      , extend.ylim.f = 0.08
      )

```

x	lm1mg1m
which	1:61:3, 5

caption	characterlistas.graphicsAnnotwhich[j]""NA
---------	---

```

panel          pointspanel.smoothadd.smooth = TRUE
sub.caption    subtitleNULLdeparse(x$call)
main           caption
ask            TRUEpar(ask=.)
...
id.n
labels.id      NULL
cex.id
qqline         qqline()
cook.levels
cook.colcook.lty

cook.legendChanges
              listNULLlegendplot.lm() list(x = "bottomleft", legend = "Cook's
              distance", lty = cook.lty, col = cook.col, text.col = cook.col,
              bty = "n", x.intersp = 1/4, y.intersp = 1/8)
add.smooth     panel
iter.smooth    iterpanel.smooth()glm
label.pos
cex.caption    caption
cex.oma.main   sub.caption
extend.ylim.f  ylim <- extendrange(r=ylim, f = *)15id.n

sub.caption
which=3 $\sqrt{|E|}|E|E$ 
which=5 $R_i/(s \times \sqrt{1-h_{ii}})h_{ii}$ influence()$hathatresiduals.glm(type = "pearson") $R[i]$ 
which=5cook.levelsaov
which=6rstandard(.) $h_{ii}$ 
glm

```

termplotlm.influencecooks.distancehatvalues

```

require(graphics)

## Analysis of the life-cycle savings data
## given in Belsley, Kuh and Welsch.
lm.SR <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
plot(lm.SR)

## 4 plots on 1 page;
## allow room for printing model formula in outer margin:
par(mfrow = c(2, 2), oma = c(0, 0, 2, 0)) -> opar
plot(lm.SR)
plot(lm.SR, id.n = NULL) # no id's
plot(lm.SR, id.n = 5, labels.id = NULL) # 5 id numbers

## Was default in R <= 2.1.x:
## Cook's distances instead of Residual-Leverage plot
plot(lm.SR, which = 1:4)

## All the above fit a smooth curve where applicable
## by default unless "add.smooth" is changed.
## Give a smoother curve by increasing the lowess span :
plot(lm.SR, panel = function(x, y) panel.smooth(x, y, span = 1))

par(mfrow = c(2,1)) # same oma as above
plot(lm.SR, which = 1:2, sub.caption = "Saving Rates, n=50, p=5")

## Cook's distance tweaking
par(mfrow = c(2,3)) # same oma ...
plot(lm.SR, which = 1:6, cook.col = "royalblue")

## A case where over plotting of the "legend" is to be avoided:
if(dev.interactive(TRUE)) getOption("device")(height = 6, width = 4)
par(mfrow = c(3,1), mar = c(5,5,4,2)/2 +.1, mgp = c(1.4, .5, 0))
plot(lm.SR, which = 5, extend.ylim.f = c(0.2, 0.08))
plot(lm.SR, which = 5, cook.lty = "dotted",
      cook.legendChanges = list(x = "bottomright", legend = "Cook"))
plot(lm.SR, which = 5, cook.legendChanges = NULL) # no "legend"

par(opar) # reset par()s

```

plot.ppr

ppr

```

## S3 method for class 'ppr'
plot(x, ask, type = "o", cex = 1/2,
      main = quote(bquote(
        "term"[.(i)]*"": " ~ hat(beta[.(i)]) == .(bet.i))),
      xlab = quote(bquote(bold(alpha)[.(i)]^T * bold(x))),
      ylab = "", ...)

```

```

x            "ppr"ppr
ask          askparTRUE
type        plot.default
cex          par("cex")
mainxlabylab titleibet.ieval()
...          plot()

```

pprpar

```

require(graphics)

rock1 <- within(rock, { area1 <- area/10000; peri1 <- peri/10000 })
par(mfrow = c(3,2)) # maybe: , pty = "s"
rock.ppr <- ppr(log(perm) ~ area1 + peri1 + shape,
               data = rock1, nterms = 2, max.terms = 5)
plot(rock.ppr, main = "ppr(log(perm)~ ., nterms=2, max.terms=5)")
plot(update(rock.ppr, bass = 5), main = "update(..., bass = 5)")
plot(update(rock.ppr, sm.method = "gcv", gcvpen = 2),
     main = "update(..., sm.method=\"gcv\", gcvpen=2)")

```

plot.profile

plotpairs"profile"

```

## S3 method for class 'profile'
plot(x, ...)
## S3 method for class 'profile'
pairs(x, colours = 2:3, which = names(x), ...)

```

```

x            "profile"
colours      xy
which
...

```

```
plotprofile.glmprofile.nlsplot.profile.nls
pairswhich
```

```
profile.glmprofile.nls
```

```
## see ?profile.glm for another example using glm fits.

## a version of example(profile.nls) from R >= 2.8.0
fm1 <- nls(demand ~ SSasympOrig(Time, A, lrc), data = BOD)
pr1 <- profile(fm1, alphamax = 0.1)
stats:::plot.profile(pr1) ## override dispatch to plot.profile.nls
pairs(pr1) # a little odd since the parameters are highly correlated

## an example from ?nls
x <- -(1:100)/10
y <- 100 + 10 * exp(x / 2) + rnorm(x)/10
nlmod <- nls(y ~ Const + A * exp(B * x), start=list(Const=100, A=10, B=1))
pairs(profile(nlmod))

## example from Dobson (1990) (see ?glm)
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
## this example is only formally a Poisson model. It is really a
## comparison of 3 multinomials. Only the interaction parameters are of
## interest.
glm.D93i <- glm(counts ~ outcome * treatment, family = poisson())
pr1 <- profile(glm.D93i)
pr2 <- profile(glm.D93i, which=6:9)
plot(pr1)
plot(pr2)
pairs(pr1)
pairs(pr2)
```

plot.profile.nls	profile.nls
------------------	-------------

```
nlsprofile.nls
```

```
## S3 method for class 'profile.nls'
plot(x, levels, conf = c(99, 95, 90, 80, 50)/100,
     absVal = TRUE, ylab = NULL, lty = 2, ...)
```

x	"profile.nls"
levels	confllevels
conf	c(0.99, 0.95, 0.90, 0.80, 0.50).
absVal	TRUE
lty	
ylab...	plot.default xlabxlimylimtype

[palette](#)

[nlsprofileprofile.nls](#)

```
require(graphics)

# obtain the fitted object
fm1 <- nls(demand ~ SSasymptOrig(Time, A, lrc), data = BOD)
# get the profile for the fitted model
pr1 <- profile(fm1, alphamax = 0.05)
opar <- par(mfrow = c(2,2), oma = c(1.1, 0, 1.1, 0), las = 1)
plot(pr1, conf = c(95, 90, 80, 50)/100)
plot(pr1, conf = c(95, 90, 80, 50)/100, absVal = FALSE)
mtext("Confidence intervals based on the profile sum of squares",
      side = 3, outer = TRUE)
mtext("BOD data - confidence levels of 50%, 80%, 90% and 95%",
      side = 1, outer = TRUE)
par(opar)
```

`plot.spec`

"spec"


```
## S3 method for class 'spec'
plot(x, add = FALSE, ci = 0.95, log = c("yes", "dB", "no"),
     xlab = "frequency", ylab = NULL, type = "l",
     ci.col = "blue", ci.lty = 3,
     main = NULL, sub = NULL,
     plot.type = c("marginal", "coherency", "phase"),
     ...)

plot.spec.phase(x, ci = 0.95,
               xlab = "frequency", ylab = "phase",
               ylim = c(-pi, pi), type = "l",
               main = NULL, ci.col = "blue", ci.lty = 3, ...)

plot.spec.coherency(x, ci = 0.95,
                   xlab = "frequency",
                   ylab = "squared coherency",
                   ylim = c(0, 1), type = "l",
                   main = NULL, ci.col = "blue", ci.lty = 3, ...)
```

x	"spec"
add	TRUEplot.type = "marginal"
ci	ci
log	"dB""yes"options(ts.S.compat = TRUE)"dB"plot.type = "marginal"
xlab	
ylab	
type	
ci.col	
ci.lty	
main	
sub	plot.type = "marginal"
plot.type	
ylim...	

[spectrum](#)

`plot.stepfun`

[plotstepfun](#)

```
## S3 method for class 'stepfun'
plot(x, xval, xlim, ylim = range(c(y, Fn.kn)),
      xlab = "x", ylab = "f(x)", main = NULL,
      add = FALSE, verticals = TRUE, do.points = (n < 1000),
      pch = par("pch"), col = par("col"),
      col.points = col, cex.points = par("cex"),
      col.hor = col, col.vert = col,
      lty = par("lty"), lwd = par("lwd"), ...)
```

```
## S3 method for class 'stepfun'
lines(x, ...)
```

```
x          "stepfun"
xval       xknots(x)xlim
xlimylim   plot.window
xlabylab
main
add         TRUE
verticals   TRUE
do.points   TRUExlim< 1000
pch         do.points
col
col.points  do.points
cex.points  do.points
col.hor
col.vert
ltylwd
...         plot(.) (add)segments(.)
```

```
t
y          t[]
```

<maechler@stat.math.ethz.ch>

[ecdfapproxfunspolinefun](#)

```

require(graphics)

y0 <- c(1,2,4,3)
sfun0 <- stepfun(1:3, y0, f = 0)
sfun.2 <- stepfun(1:3, y0, f = .2)
sfun1 <- stepfun(1:3, y0, right = TRUE)

tt <- seq(0, 3, by = 0.1)
op <- par(mfrow = c(2,2))
plot(sfun0); plot(sfun0, xval = tt, add = TRUE, col.hor = "bisque")
plot(sfun.2); plot(sfun.2, xval = tt, add = TRUE, col = "orange") # all colors
plot(sfun1); lines(sfun1, xval = tt, col.hor = "coral")
##-- This is revealing :
plot(sfun0, verticals = FALSE,
      main = "stepfun(x, y0, f=f) for f = 0, .2, 1")
for(i in 1:3)
  lines(list(sfun0, sfun.2, stepfun(1:3, y0, f = 1))[[i]], col = i)
legend(2.5, 1.9, paste("f =", c(0, 0.2, 1)), col = 1:3, lty = 1, y.intersp = 1)
par(op)

# Extend and/or restrict 'viewport':
plot(sfun0, xlim = c(0,5), ylim = c(0, 3.5),
      main = "plot(stepfun(*), xlim= . , ylim = .)")

##-- this works too (automatic call to ecdf(.)):
plot.stepfun(rt(50, df = 3), col.vert = "gray20")

```

plot.ts

```

"ts"

## S3 method for class 'ts'
plot(x, y = NULL, plot.type = c("multiple", "single"),
      xy.labels, xy.lines, panel = lines, nc, yax.flip = FALSE,
      mar.multi = c(0, 5.1, 0, if(yax.flip) 5.1 else 2.1),
      oma.multi = c(6, 0, 5, 0), axes = TRUE, ...)

## S3 method for class 'ts'
lines(x, ...)

xy          "ts"
plot.type
xy.labels   text()
xy.lines    linesxy.labelsTRUE
panel       function(x, col, bg, pch, type, ...)plot.type = "multiple"lines
nc          type = "multiple"

```

```

yax.flip          type = "multiple"
mar.multioma.multi
                  parplot.type = "multiple"

axes
...              plotplot.defaultpar

yplot.type
xyxy ~ xtextxy.labelsTRUEcharacterlinesxy.linesTRUE

```

ts

```

require(graphics)

## Multivariate
z <- ts(matrix(rt(200 * 8, df = 3), 200, 8),
          start = c(1961, 1), frequency = 12)
plot(z, yax.flip = TRUE)
plot(z, axes = FALSE, ann = FALSE, frame.plot = TRUE,
      mar.multi = c(0,0,0,0), oma.multi = c(1,1,5,1))
title("plot(ts(..), axes=FALSE, ann=FALSE, frame.plot=TRUE, mar..., oma...)")

z <- window(z[,1:3], end = c(1969,12))
plot(z, type = "b")    # multiple
plot(z, plot.type = "single", lty = 1:3, col = 4:2)

## A phase plot:
plot(nhtemp, lag(nhtemp, 1), cex = .8, col = "blue",
     main = "Lag plot of New Haven temperatures")

## xy.lines and xy.labels are FALSE for large series:
plot(lag(sunspots, 1), sunspots, pch = ".")

SMI <- EuStockMarkets[, "SMI"]
plot(lag(SMI, 1), SMI, pch = ".")
plot(lag(SMI, 20), SMI, pch = ".", log = "xy",
     main = "4 weeks lagged SMI stocks -- log scale", xy.lines = TRUE)

```

Poisson

lambda

```

dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)

```

`x`
`q`
`p`
`n`
`lambda`
`loglog.p`
`lower.tail` $P[X \leq x]P[X > x]$

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

$x = 0, 1, 2, \dots$
 $E(X) = Var(X) = \lambda$
 $\lambda = 00^0 = 10$
`xdpois` $p(x)$ `dbinom`
`qpois`(`p`, `lambda`) $xP(X \leq x) \geq p$
`lower.tail = FALSE``lower.tail = TRUE`

`dpois``ppois``qpois``rpois`
`lambda``NaN`
`nrpois`
`n`
`rpois``double`

`dpois``dbinom`
`ppois``pgamma`
`qpois`
`rpois`

`dbinom``dnbinom`
`poisson.test`

```

require(graphics)

-log(dpois(0:7, lambda = 1) * gamma(1+ 0:7)) # == 1
Ni <- rpois(50, lambda = 4); table(factor(Ni, 0:max(Ni)))

1 - ppois(10*(15:25), lambda = 100) # becomes 0 (cancellation)
  ppois(10*(15:25), lambda = 100, lower.tail = FALSE) # no cancellation

par(mfrow = c(2, 1))
x <- seq(-0.01, 5, 0.01)
plot(x, ppois(x, 1), type = "s", ylab = "F(x)", main = "Poisson(1) CDF")
plot(x, pbinom(x, 100, 0.01), type = "s", ylab = "F(x)",
      main = "Binomial(100, 0.01) CDF")

## The (limit) case lambda = 0 :
stopifnot(identical(dpois(0,0), 1),
           identical(ppois(0,0), 1),
           identical(qpois(1,0), 0))

```

poisson.test

```

poisson.test(x, T = 1, r = 1,
             alternative = c("two.sided", "less", "greater"),
             conf.level = 0.95)

```

```

x
T
r
alternative      "two.sided" "greater" "less"
conf.level

```

[binom.test](#)[binom.test](#)

```

"htest"
statistic
parameter
p.value
conf.int
estimate
null.value      r
alternative
method          "Exact Poisson test" "Comparison of Poisson rates"
data.name

```

T
n

[binom.test](#)

```
### These are paraphrased from data sets in the ISwR package

## SMR, Welsh Nickel workers
poisson.test(137, 24.19893)

## eba1977, compare Fredericia to other three cities for ages 55-59
poisson.test(c(11, 6+8+7), c(800, 1083+1050+878))
```

poly

degree

```
poly(x, ..., degree = 1, coefs = NULL, raw = FALSE, simple = FALSE)
polym (..., degree = 1, coefs = NULL, raw = FALSE)
```

```
## S3 method for class 'poly'
predict(object, newdata, ...)
```

xnewdata	mode "numeric" Date x
degree	raw
coefs	
raw	
simple	attributes dimnames
object	"poly"poly
...	polypolym
	predict.poly

```
degree...poly(x, 3)
predict
poly...polymcoefpolym...poly
formulasubset = *xpoly(x)model.frame
```

```
polypolym()simple=FALSEcoefs=NULL
x"degree"raw = TRUE"coefs"c("poly", "matrix")
poly(*, simple=TRUE)polym(*, coefs=<non-NULL>)predict.poly()
```

contr.poly

contr.poly

cars

```
od <- options(digits = 3) # avoid too much visual clutter
(z <- poly(1:10, 3))
predict(z, seq(2, 4, 0.5))
zapsmall(poly(seq(4, 6, 0.5), 3, coefs = attr(z, "coefs")))

zm <- zapsmall(polym ( 1:4, c(1, 4:6), degree = 3)) # or just poly():
(z1 <- zapsmall(poly(cbind(1:4, c(1, 4:6)), degree = 3)))
## they are the same :
stopifnot(all.equal(zm, z1, tolerance = 1e-15))

## poly(<matrix>, df) --- used to fail till July 14 (vive la France!), 2017:
m2 <- cbind(1:4, c(1, 4:6))
pm2 <- zapsmall(poly(m2, 3)) # "unnamed degree = 3"
stopifnot(all.equal(pm2, zm, tolerance = 1e-15))

options(od)
```

power

$$\eta = \mu^\lambda$$

power(lambda = 1)

lambda

lambdalambda = 1

linkfunlinkinvmu.etavalidetamake.link

[make.linkfamily](#)

[Arithmetic](#)

[power.t.test](#)

```
power()  
quasi(link = power(1/3))[c("linkfun", "linkinv")]
```

[power.anova.test](#)

```
power.anova.test(groups = NULL, n = NULL,  
                 between.var = NULL, within.var = NULL,  
                 sig.level = 0.05, power = NULL)
```

groups

n

between.var

within.var

sig.level

power

groupsnbetween.varpowerwithin.varsig.levelsig.level

"power.htest"methodnote

uniroot

[anova](#)[lm](#)[uniroot](#)

```

power.anova.test(groups = 4, n = 5, between.var = 1, within.var = 3)
# Power = 0.3535594

power.anova.test(groups = 4, between.var = 1, within.var = 3,
                  power = .80)
# n = 11.92613

## Assume we have prior knowledge of the group means:
groupmeans <- c(120, 130, 140, 150)
power.anova.test(groups = length(groupmeans),
                  between.var = var(groupmeans),
                  within.var = 500, power = .90) # n = 15.18834

```

power.prop.test

```

power.prop.test(n = NULL, p1 = NULL, p2 = NULL, sig.level = 0.05,
                power = NULL,
                alternative = c("two.sided", "one.sided"),
                strict = FALSE, tol = .Machine$double.eps^0.25)

```

```

n
p1
p2
sig.level
power
alternative
strict
tol

```

```

np1p2powersig.levelsig.levelNULL
strict = TRUE

```

```

power.prop.test(n=30, p1=0.90, p2=NULL, power=0.8, strict=TRUE)

```

```

p2p1 = 0.9n = 74(p1,p2) = (0.9,1)

```

[warningstop](#)

"power.htest"methodnote

`uniroot` $p_1 < p_2 \leq p_1$

`prop.test``uniroot`

```
power.prop.test(n = 50, p1 = .50, p2 = .75)      ## => power = 0.740
power.prop.test(p1 = .50, p2 = .75, power = .90) ## =>      n = 76.7
power.prop.test(n = 50, p1 = .5, power = .90)    ## =>      p2 = 0.8026
power.prop.test(n = 50, p1 = .5, p2 = 0.9, power = .90, sig.level=NULL)
                                                ## => sig.l = 0.00131
power.prop.test(p1 = .5, p2 = 0.501, sig.level=.001, power=0.90)
                                                ## => n = 10451937

try(
  power.prop.test(n=30, p1=0.90, p2=NULL, power=0.8)
) # a warning (which may become an error)
## Reason:
power.prop.test(      p1=0.90, p2= 1.0, power=0.8) ##-> n = 73.37
```

`power.t.test`

```
power.t.test(n = NULL, delta = NULL, sd = 1, sig.level = 0.05,
             power = NULL,
             type = c("two.sample", "one.sample", "paired"),
             alternative = c("two.sided", "one.sided"),
             strict = FALSE, tol = .Machine$double.eps^0.25)
```

```
n
delta
sd
sig.level
power
type
alternative
strict
tol
```

```
ndeltapowersdsig.level=NULL
strict = TRUE
```

```
"power.htest"methodnote
```

```
uniroot
```

```
t.testuniroot
```

```
power.t.test(n = 20, delta = 1)
power.t.test(power = .90, delta = 1)
power.t.test(power = .90, delta = 1, alternative = "one.sided")
```

```
PP.test
```

```
x
```

```
PP.test(x, lshort = TRUE)
```

```
x
```

```
lshort
```

```
sigma^2lshortTRUEtrunc(4*(n/100)^0.25)trunc(12*(n/100)^0.25)
```

```
"htest"
```

```
statistic
```

```
parameter
```

```
p.value
```

```
method
```

```
data.name
```

```

x <- rnorm(1000)
PP.test(x)
y <- cumsum(x) # has unit root
PP.test(y)

```

ppoints

```
(1:m - a)/(m + (1-a)-a)mnlength(n)==1length(n)
```

```
ppoints(n, a = if(n <= 10) 3/8 else 1/2)
```

```

n
a          (0,1)

```

```
0 < a < 1(0,1)[0,1]p + rev(p) == 1
```

```
ppoints()qqplotqqnorm
```

```
aquantile
```

```
quantilea
```

```
qqplotqqnorm
```

```

ppoints(4) # the same as ppoints(1:4)
ppoints(10)
ppoints(10, a = 1/2)

## Visualize including the fractions :
require(graphics)
p.ppoints <- function(n, ..., add = FALSE, col = par("col")) {
  pn <- ppoints(n, ...)
  if(add)
    points(pn, pn, col = col)
  else {
    tit <- match.call(); tit[[1]] <- quote(ppoints)
    plot(pn,pn, main = deparse(tit), col=col,
         xlim = 0:1, ylim = 0:1, xaxs = "i", yaxs = "i")
    abline(0, 1, col = adjustcolor(1, 1/4), lty = 3)
  }
  if(!add && requireNamespace("MASS", quietly = TRUE))
    text(pn, pn, as.character(MASS::fractions(pn)),
         adj = c(0,0)-1/4, cex = 3/4, xpd = NA, col=col)
  abline(h = pn, v = pn, col = adjustcolor(col, 1/2), lty = 2, lwd = 1/2)
}

p.ppoints(4)
p.ppoints(10)
p.ppoints(10, a = 1/2)
p.ppoints(21)
p.ppoints(8) ; p.ppoints(8, a = 1/2, add=TRUE, col="tomato")

```

ppr

```

ppr(x, ...)

## S3 method for class 'formula'
ppr(formula, data, weights, subset, na.action,
     contrasts = NULL, ..., model = FALSE)

## Default S3 method:
ppr(x, y, weights = rep(1, n),
     ww = rep(1, q), nterms, max.terms = nterms, optlevel = 2,
     sm.method = c("supsmu", "spline", "gcv spline"),
     bass = 0, span = 0, df = 5, gcvpen = 1, trace = FALSE, ...)

```

formula

x

y

nterms	
data	<code>model.frame</code> formula
weights	<code>w_i</code>
ww	<code>ijw_i ww_j (y_ij - fit_ij)^2w_i</code>
subset	
na.action	<code>NAgetOption("na.action")</code>
contrasts	
max.terms	
optlevel	
sm.method	<code>supsmusmooth.spline</code>
bass	<code>supsmu</code>
span	<code>supsmu0span(0, 1]</code>
df	<code>sm.method"spline"</code>
gcvpen	<code>sm.method"gcv spline"</code>
trace	<code>lambda d f supsmu</code>
...	
model	

max.terms	nterms
optlevel	

call	
p	
q	
mu	nterms
ml	max.terms
gof	
gofn	max.termsnterms
df	df
edf	sm.method"spline""gcv spline"
xnames	
yname	
alpha	
beta	
yb	
ys	ys
fitted.values	q > 1
residuals	q > 1
smod	
model	model = TRUE

[plot.pprsupsmusmooth.spline](#)

```
require(graphics)

# Note: your numerical values may differ
attach(rock)
area1 <- area/10000; peri1 <- peri/10000
rock.ppr <- ppr(log(perm) ~ area1 + peri1 + shape,
               data = rock, nterms = 2, max.terms = 5)

rock.ppr
# Call:
# ppr.formula(formula = log(perm) ~ area1 + peri1 + shape, data = rock,
#             nterms = 2, max.terms = 5)
#
# Goodness of fit:
# 2 terms 3 terms 4 terms 5 terms
# 8.737806 5.289517 4.745799 4.490378

summary(rock.ppr)
# ..... (same as above)
# .....
#
# Projection direction vectors ('alpha'):
#      term 1      term 2
# area1  0.34357179  0.37071027
# peri1 -0.93781471 -0.61923542
# shape  0.04961846  0.69218595
#
# Coefficients of ridge terms:
#      term 1      term 2
# 1.6079271  0.5460971

par(mfrow = c(3,2)) # maybe: , pty = "s")
plot(rock.ppr, main = "ppr(log(perm)~ ., nterms=2, max.terms=5)")
plot(update(rock.ppr, bass = 5), main = "update(..., bass = 5)")
plot(update(rock.ppr, sm.method = "gcv", gcvpen = 2),
     main = "update(..., sm.method=\"gcv\", gcvpen=2)")
cbind(perm = rock$perm, prediction = round(exp(predict(rock.ppr)), 1))
detach()
```

prcomp

prcomp

prcomp(x, ...)

```
## S3 method for class 'formula'
prcomp(formula, data = NULL, subset, na.action, ...)
```

```
## Default S3 method:
prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE,
       tol = NULL, rank. = NULL, ...)
```

```
## S3 method for class 'prcomp'
predict(object, newdata, ...)
```

formula

data [model.frame](#)formulaenvironment(formula)

subset x

na.action NAna.actionoptionsna.failna.omit

... xscale.tol

x

retx

center xscale

scale. FALSEx[scale](#)

tol tolrank.min(dim(x))toltol = 0tol = sqrt(.Machine\$double.eps)

rank. tol

object "prcomp"

newdata newdata

eigenprintplot

[princomp](#) $N - 1$

scale = TRUEcenter = TRUE

prcomp"prcomp"

sdev

rotation princomploadings

x retxrotationcov(x)diag(sdev^2)[napredict\(\)](#)na.action

centerscale FALSE

biplot.prcompscreeplotprincompcorcovsvdeigen

```
C <- chol(S <- toeplitz(.9 ^ (0:31))) # Cov.matrix and its root
all.equal(S, crossprod(C))
set.seed(17)
X <- matrix(rnorm(32000), 1000, 32)
Z <- X %*% C ## ==> cov(Z) ~= C'C = S
all.equal(cov(Z), S, tolerance = 0.08)
pZ <- prcomp(Z, tol = 0.1)
summary(pZ) # only ~14 PCs (out of 32)
## or choose only 3 PCs more directly:
pz3 <- prcomp(Z, rank. = 3)
summary(pz3) # same numbers as the first 3 above
stopifnot(ncol(pZ$rotation) == 14, ncol(pz3$rotation) == 3,
          all.equal(pz3$sdev, pZ$sdev, tolerance = 1e-15)) # exactly equal typically

## signs are random
require(graphics)
## the variances of the variables in the
## USArrests data vary by orders of magnitude, so scaling is appropriate
prcomp(USArrests) # inappropriate
prcomp(USArrests, scale. = TRUE)
prcomp(~ Murder + Assault + Rape, data = USArrests, scale. = TRUE)
plot(prcomp(USArrests))
summary(prcomp(USArrests, scale. = TRUE))
biplot(prcomp(USArrests, scale. = TRUE))
```

predict

predictclass

predict(object, ...)

object

...

newdata=newdata

n.ahead

se.fit

predict

[predict.glm](#)[predict.lm](#)[predict.loess](#)[predict.nls](#)[predict.poly](#)[predict.princomp](#)
[predict.smooth.spline](#)

[predict.ar](#)[predict.Arima](#)[predict.arma](#)[predict.HoltWinters](#)[predict.StructTS](#)

require(utils)

All the "predict" methods found

NB most of the methods in the standard packages are hidden.

Output will depend on what namespaces are (or have been) loaded.

for(fn in methods("predict"))

 try({

 f <- eval(substitute(getAnywhere(fn)\$objs[[1]]), list(fn = fn))

 cat(fn, ":\n\t", deparse(args(f)), "\n")

 }, silent = TRUE)

predict.Arima

[arima](#)

S3 method for class 'Arima'

predict(object, n.ahead = 1, newxreg = NULL,

 se.fit = TRUE, ...)

object arima

n.ahead

newxreg xregn.ahead

se.fit

...

[KalmanForecast](#)`predict.Arima`

`se.fit = TRUE``predse`

[arima](#)

```
od <- options(digits = 5) # avoid too much spurious accuracy
predict(arima(lh, order = c(3,0,0)), n.ahead = 12)

(fit <- arima(USAccDeaths, order = c(0,1,1),
              seasonal = list(order = c(0,1,1))))
predict(fit, n.ahead = 6)
options(od)
```

`predict.glm`

```
## S3 method for class 'glm'
predict(object, newdata = NULL,
        type = c("link", "response", "terms"),
        se.fit = FALSE, dispersion = NULL, terms = NULL,
        na.action = na.pass, ...)
```

```
object      "glm"
newdata
type        "response" type = "response" "terms"
```

```
se.fit
dispersion  summary
terms       type = "terms"
na.action   newdataNA
...
```

```
newdata=na.action=na.omitna.action=na.excludeNApredict
```

```
se.fit = FALSEtype = "terms""constant"
```

```
se.fit = TRUE
```

```
fit          se.fit = FALSE
```

```
se.fit
```

```
residual.scale
```

```
newdata=newdata
```

[glmSafePrediction](#)

```
require(graphics)
```

```
## example from Venables and Ripley (2002, pp. 190-2.)
```

```
ldose <- rep(0:5, 2)
```

```
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
```

```
sex <- factor(rep(c("M", "F"), c(6, 6)))
```

```
SF <- cbind(numdead, numalive = 20-numdead)
```

```
budworm.lg <- glm(SF ~ sex*ldose, family = binomial)
```

```
summary(budworm.lg)
```

```
plot(c(1,32), c(0,1), type = "n", xlab = "dose",
```

```
      ylab = "prob", log = "x")
```

```
text(2^ldose, numdead/20, as.character(sex))
```

```
ld <- seq(0, 5, 0.1)
```

```
lines(2^ld, predict(budworm.lg, data.frame(ldose = ld,
```

```
      sex = factor(rep("M", length(ld)), levels = levels(sex))),
```

```
      type = "response"))
```

```
lines(2^ld, predict(budworm.lg, data.frame(ldose = ld,
```

```
      sex = factor(rep("F", length(ld)), levels = levels(sex))),
```

```
      type = "response"))
```

```
predict.HoltWinters
```

```
## S3 method for class 'HoltWinters'
```

```
predict(object, n.ahead = 1, prediction.interval = FALSE,
```

```
      level = 0.95, ...)
```

```
object      HoltWinters
n.ahead
prediction.interval
            TRUE
level
...
```

```
fitlwrupr
```

```
<David.Meyer@wu.ac.at>
```

[HoltWinters](#)

```
require(graphics)

m <- HoltWinters(co2)
p <- predict(m, 50, prediction.interval = TRUE)
plot(m, p)
```

```
predict.lm
```

```
## S3 method for class 'lm'
predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
        interval = c("none", "confidence", "prediction"),
        level = 0.95, type = c("response", "terms"),
        terms = NULL, na.action = na.pass,
        pred.var = res.var/weights, weights = 1,
        rankdeficient = c("warnif", "simple", "non-estim", "NA", "NAwarn"),
        tol = 1e-6, verbose = FALSE,
        ...)
```

```

object      "lm"
newdata
se.fit
scale
df
interval
level
type
terms      type = "terms"character
na.action  newdataNA
pred.var
weights    newdata
rankdeficient characterobject$rank < ncol(model.matrix(object))
           "warnif" warningtol"non-estim"rankdeficient="non-estim"
           "simple" warning
           "non-estim" warningattr(*, "non-estim")1:nrow(newdata)
           "NA" NA
           "NAwarn" NAwarning
tol
verbose    logical
...

```

```

predict.lmnewdatamodel.frame(object)se.fitTRUEscaledfintervalslevel
lmcoef()NAnewdatanewdatanon-estimabletolrankdeficient == "simple"NA
rankdeficient == "warnif"tolrankdeficientNA
newdatana.actionna.action = na.omitna.action = na.excludeNApredict
newdatapred.varres.var $\sigma^2$ weightsweightsnewdata

```

```

predict.lmfitlwruprintervaltype = "terms""constant"
se.fitTRUE
fit
se.fit
residual.scale
df

```

```

newdatanewdata

```

```

res.var

```

lmpredict

```
require(graphics)

## Predictions
x <- rnorm(15)
y <- x + rnorm(15)
predict(lm(y ~ x))
new <- data.frame(x = seq(-3, 3, 0.5))
predict(lm(y ~ x), new, se.fit = TRUE)
pred.w.plim <- predict(lm(y ~ x), new, interval = "prediction")
pred.w.clim <- predict(lm(y ~ x), new, interval = "confidence")
matplot(new$x, cbind(pred.w.clim, pred.w.plim[, -1]),
        lty = c(1, 2, 2, 3, 3), type = "l", ylab = "predicted y")

## Prediction intervals, special cases
## The first three of these throw warnings
w <- 1 + x^2
fit <- lm(y ~ x)
wfit <- lm(y ~ x, weights = w)
predict(fit, interval = "prediction")
predict(wfit, interval = "prediction")
predict(wfit, new, interval = "prediction")
predict(wfit, new, interval = "prediction", weights = (new$x)^2)
predict(wfit, new, interval = "prediction", weights = ~x^2)

##-- From aov(.) example ---- predict(.. terms)
npk.aov <- aov(yield ~ block + N*P*K, npk)
(termL <- attr(terms(npk.aov), "term.labels"))
(pt <- predict(npk.aov, type = "terms"))
pt. <- predict(npk.aov, type = "terms", terms = termL[1:4])
stopifnot(all.equal(pt[, 1:4], pt.,
                    tolerance = 1e-12, check.attributes = FALSE))
```

predict.loess

loess

```
## S3 method for class 'loess'
predict(object, newdata = NULL, se = FALSE,
        na.action = na.pass, ...)
```

```
object      loess
newdata
se
na.action    newdataNA
...
```



```
se = TRUE  $N \times N$   $f f$  = spanse = TRUE  $O(N^2)$   $N$  span = 0.75
surface = "interpolate" predict.loessNA
```

```
se = FALSE newdatase = TRUE
```

```
fit
```

```
se
```

```
residual.scale
```

```
df
```

```
newdata expand.grid
```

```
NA loess
```

```
newdata newdata
```

```
cloess
```

```
loess
```

```
cars.lo <- loess(dist ~ speed, cars)
predict(cars.lo, data.frame(speed = seq(5, 30, 1)), se = TRUE)
# to get extrapolation
cars.lo2 <- loess(dist ~ speed, cars,
  control = loess.control(surface = "direct"))
predict(cars.lo2, data.frame(speed = seq(5, 30, 1)), se = TRUE)
```

```
predict.nls
```

```
predict.nls newdatase.fit TRUE scaledf intervals level
```

```
se.fit interval
```

```
## S3 method for class 'nls'
```

```
predict(object, newdata , se.fit = FALSE, scale = NULL, df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95, ...)
```

```
object      nls
newdata     newdata
se.fit      FALSE
scale       df
df          scale
interval
level
...
```

```
predict.nlsintervalfitlwruprse.fitTRUE
```

```
fit
se.fit
residual.scale
df
```

```
newdatanewdata
```

[nlspredict](#)

```
require(graphics)

fm <- nls(demand ~ SSasymptOrig(Time, A, lrc), data = BOD)
predict(fm) # fitted values at observed times
## Form data plot and smooth line for the predictions
opar <- par(las = 1)
plot(demand ~ Time, data = BOD, col = 4,
     main = "BOD data and fitted first-order curve",
     xlim = c(0,7), ylim = c(0, 20) )
tt <- seq(0, 8, length.out = 101)
lines(tt, predict(fm, list(Time = tt)))
par(opar)
```

`predict.smooth.spline`

```
## S3 method for class 'smooth.spline'
predict(object, x, deriv = 0, ...)
```

```
object          smooth.spline
x
deriv
...
```

```
x          x
y          x
```

[smooth.spline](#)

```
require(graphics)
```

```
attach(cars)
cars.spl <- smooth.spline(speed, dist, df = 6.4)
```

```
## "Proof" that the derivatives are okay, by comparing with approximation
diff.quot <- function(x, y) {
  ## Difference quotient (central differences where available)
  n <- length(x); i1 <- 1:2; i2 <- (n-1):n
  c(diff(y[i1]) / diff(x[i1]), (y[-i1] - y[-i2]) / (x[-i1] - x[-i2]),
    diff(y[i2]) / diff(x[i2]))
}
```

```
xx <- unique(sort(c(seq(0, 30, by = .2), kn <- unique(speed))))
i.kn <- match(kn, xx) # indices of knots within xx
op <- par(mfrow = c(2,2))
plot(speed, dist, xlim = range(xx), main = "Smooth.spline & derivatives")
lines(pp <- predict(cars.spl, xx), col = "red")
points(kn, pp$y[i.kn], pch = 3, col = "dark red")
mtext("s(x)", col = "red")
for(d in 1:3){
  n <- length(pp$x)
  plot(pp$x, diff.quot(pp$x,pp$y), type = "l", xlab = "x", ylab = "",
    col = "blue", col.main = "red",
```

```

      main = paste0("s" ,paste(rep("'", d), collapse = ""), "(x)")
    mtext("Difference quotient approx.(last)", col = "blue")
    lines(pp <- predict(cars.spl, xx, deriv = d), col = "red")

    points(kn, pp$y[i.kn], pch = 3, col = "dark red")
    abline(h = 0, lty = 3, col = "gray")
  }
  detach(); par(op)

```

preplot

```
preplot(object, ...)
```

object

...

object

princomp

```
princompprincomp
```

```
princomp(x, ...)
```

```
## S3 method for class 'formula'
princomp(formula, data = NULL, subset, na.action, ...)
```

```
## Default S3 method:
princomp(x, cor = FALSE, scores = TRUE, covmat = NULL,
         subset = rep_len(TRUE, nrow(as.matrix(x))), fix_sign = TRUE, ...)
```

```
## S3 method for class 'princomp'
predict(object, newdata, ...)
```

```

formula
data          model.frameformulaenvironment(formula)
subset        x
na.action     NAna.actionoptionsna.failna.omit
x
cor
scores
covmat        cov.wtcov.mvecov.mcdx
fix_sign
...           xcorscores
object        "princomp"
newdata       newdata

```

```

princomp"formula""default"
eigencorsvdxprcomp
N
printplotsscreeplotbiplot
xnapredict
princompprcomp

```

```

princomp"princomp"
sdev
loadings      "loadings"loadingsprint
center
scale
n.obs
scores        scores = TRUExcovmatnapredict()na.action
call
na.action

```

```

fix_sign = TRUE

```

```

summary.princompscreeplotbiplot.princompprcompcorcoveigen

```

```

require(graphics)

## The variances of the variables in the
## USArrests data vary by orders of magnitude, so scaling is appropriate
(pc.cr <- princomp(USArrests)) # inappropriate
princomp(USArrests, cor = TRUE) # =^= prcomp(USArrests, scale=TRUE)
## Similar, but different:
## The standard deviations differ by a factor of sqrt(49/50)

summary(pc.cr <- princomp(USArrests, cor = TRUE))
loadings(pc.cr) # note that blank entries are small but not zero
## The signs of the columns of the loadings are arbitrary
plot(pc.cr) # shows a screeplot.
biplot(pc.cr)

## Formula interface
princomp(~ ., data = USArrests, cor = TRUE)

## NA-handling
USArrests[1, 2] <- NA
pc.cr <- princomp(~ Murder + Assault + UrbanPop,
                  data = USArrests, na.action = na.exclude, cor = TRUE)
pc.cr$scores[1:5, ]

## (Simple) Robust PCA:
## Classical:
(pc.cl <- princomp(stackloss))
## Robust:
(pc.rob <- princomp(stackloss, covmat = MASS::cov.rob(stackloss)))

```

```
print.power.htest
```

```
"htest""power.htest"print
```

```
## S3 method for class 'htest'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

```
## S3 method for class 'power.htest'
print(x, digits = getOption("digits"), ...)
```

```

x                "htest""power.htest"
digits
prefix           strwrapmethodhtest
...

```

```

printdigitshtestmax(1, digits - 2)
power.htestmethodnotemethodnote

```

xprint

power.t.testpower.prop.test

```
(ptt <- power.t.test(n = 20, delta = 1))
print(ptt, digits = 4) # using less digits than default
print(ptt, digits = 12) # using more " " "
```

print.ts

formatprint

```
## S3 method for class 'ts'
print(x, calendar, ...)
.preformat.ts(x, calendar, ...)
```

```
x
calendar      TRUEFALSE
...           printformat
```

print"ts"tsp(x)calendar.preformat.ts(x, *)matrixrownames

printts

```
print(ts(1:10, frequency = 7, start = c(12, 2)), calendar = TRUE)

print(sunsp.1 <- window(sunspot.m2014, end=c(1756, 12)))
m <- .preformat.ts(sunsp.1) # a character matrix
```

printCoefmat

[printsummary.lmsummary.glmanovax](#)

```
printCoefmat(x, digits = max(3, getOption("digits") - 2),
             signif.stars = getOption("show.signif.stars"),
             signif.legend = signif.stars,
             dig.tst = max(1, min(5, digits - 1)),
             cs.ind = 1L:k, tst.ind = k + 1L,
             zap.ind = integer(), P.values = NULL,
             has.Pvalue = nc >= 4L && length(cn <- colnames(x)) &&
               substr(cn[nc], 1L, 3L) %in% c("Pr(", "p-v"),
             eps.Pvalue = .Machine$double.eps,
             na.print = "NA", quote = FALSE, right = TRUE, ...)
```

```
x
digits
signif.stars    TRUEshow.signif.starsoptions
signif.legend   TRUEsignif.stars = TRUE
dig.tst         tst.ind
cs.ind
tst.ind
zap.ind         zapsmall
P.values        NULLTRUEExformat.pvalP.values      =      NULLTRUE
options("show.coef.Pvalue")TRUEEx"Pr("
has.Pvalue      TRUEExP.values
eps.Pvalue      format.pval()eps
na.print        NA
quoteright...   print.default
```

x

[print.summary.lmformat.pvalformat](#)


```

cmat <- cbind(rnorm(3, 10), sqrt(rchisq(3, 12)))
cmat <- cbind(cmat, cmat[, 1]/cmat[, 2])
cmat <- cbind(cmat, 2*pnorm(-cmat[, 3]))
colnames(cmat) <- c("Estimate", "Std.Err", "Z value", "Pr(>z)")
printCoefmat(cmat[, 1:3])
printCoefmat(cmat)
op <- options(show.coef.Pvalues = FALSE)
printCoefmat(cmat, digits = 2)
printCoefmat(cmat, digits = 2, P.values = TRUE)
options(op) # restore

```

profile

fitted

profile(fitted, ...)

fitted

...

[profile.nlsprofile.glm](#)

[plot.profile](#)

[Rprof](#)

profile.glm

glm

"glm"

```
## S3 method for class 'glm'
```

```

profile(fitted, which = 1:p, alpha = 0.01, maxsteps = 10,
        del = zmax/5, trace = FALSE, test = c("LRT", "Rao"), ...)

```

```
fitted
which
alpha
maxsteps
del
trace
test
...
```

```
test = "LRT"test = "Rao"
```

```
"profile.glm""profile"
par.vals
tau or z
```

[glmprofileplot.profile](#)

```
options(contrasts = c("contr.treatment", "contr.poly"))
ldose <- rep(0:5, 2)
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"), c(6, 6)))
SF <- cbind(numdead, numalive = 20 - numdead)
budworm.lg <- glm(SF ~ sex*ldose, family = binomial)
pr1 <- profile(budworm.lg)
plot(pr1)
pairs(pr1)
```

profile.nls

nls

"nls"

```
## S3 method for class 'nls'
profile(fitted, which = 1:npar, maxpts = 100, alphamax = 0.01,
        delta.t = cutoff/5, ...)
```

```
fitted
which
maxpts
alphamax
delta.t
...
```

```
par.vals
tau
```

[nlsprofileplot.profile.nls](#)

```
# obtain the fitted object
fm1 <- nls(demand ~ SSasympOrig(Time, A, lrc), data = BOD)
# get the profile for the fitted model: default level is too extreme
pr1 <- profile(fm1, alphamax = 0.05)
# profiled values for the two parameters

pr1$A
pr1$lrc

# see also example(plot.profile.nls)
```

proj

[proj](#)[aov](#)

`proj(object, ...)`

S3 method for class 'aov'

`proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)`

S3 method for class 'aovlist'

`proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)`

Default S3 method:

`proj(object, onedf = TRUE, ...)`

S3 method for class 'lm'

`proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)`

`object` "lm"qreffects

`onedf` TRUEFALSE

`unweighted.scale`

`object`

...

`aovError`

`onedf = FALSE``onedf = TRUE`

`lmaov`

`onedf = FALSE`

[aovlmmodel.tables](#)

```

N <- c(0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,0,1,0,1,1,0,0)
P <- c(1,1,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,0,0,1,0,1,1,0)
K <- c(1,0,0,1,0,1,1,0,0,1,0,1,0,1,1,0,0,0,1,1,1,0,1,0)
yield <- c(49.5,62.8,46.8,57.0,59.8,58.5,55.5,56.0,62.8,55.8,69.5,
55.0, 62.0,48.8,45.5,44.2,52.0,51.5,49.8,48.8,57.2,59.0,53.2,56.0)

npk <- data.frame(block = gl(6,4), N = factor(N), P = factor(P),
                  K = factor(K), yield = yield)
npk.aov <- aov(yield ~ block + N*P*K, npk)
proj(npk.aov)

## as a test, not particularly sensible
options(contrasts = c("contr.helmert", "contr.treatment"))
npk.aovE <- aov(yield ~ N*P*K + Error(block), npk)
proj(npk.aovE)

```

prop.test

prop.test

```

prop.test(x, n, p = NULL,
          alternative = c("two.sided", "less", "greater"),
          conf.level = 0.95, correct = TRUE)

```

```

x
n          x
p          px
alternative "two.sided""greater""less"
conf.level
correct

```

```

pNULLalternativeconf.level[-1,1]"two.sided"NULL
pppalternativeconf.level[0,1]
pp"two.sided"NULL

```

```

"htest"

statistic
parameter
p.value

```

```
estimate      x/n
conf.int      pNULLNULLconf.level
null.value    pNULL
alternative
method
data.name
```

`binom.test`

```
heads <- rbinom(1, size = 100, prob = .5)
prop.test(heads, 100)      # continuity correction TRUE by default
prop.test(heads, 100, correct = FALSE)

## Data from Fleiss (1981), p. 139.
## H0: The null hypothesis is that the four populations from which
##      the patients were drawn have the same true proportion of smokers.
## A:  The alternative is that this proportion is different in at
##      least one of the populations.

smokers <- c( 83, 90, 129, 70 )
patients <- c( 86, 93, 136, 82 )
prop.test(smokers, patients)
```

`prop.trend.test`

score

```
prop.trend.test(x, n, score = seq_along(x))
```

x

n

score

"htest"

prop.test

prop.test

```
smokers <- c( 83, 90, 129, 70 )
patients <- c( 86, 93, 136, 82 )
prop.test(smokers, patients)
prop.trend.test(smokers, patients)
prop.trend.test(smokers, patients, c(0,0,0,1))
```

qqnorm

```
qqnormqqlineprobs
qqplotconf.levelxyexact = NULLbsimulate = TRUE
qqnormqqplotqqline

qqnorm(y, ...)
## Default S3 method:
qqnorm(y, ylim, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       plot.it = TRUE, datax = FALSE, ...)

qqline(y, datax = FALSE, distribution = qnorm,
       probs = c(0.25, 0.75), qtype = 7, ...)

qqplot(x, y, plot.it = TRUE,
       xlab = deparse1(substitute(x)),
       ylab = deparse1(substitute(y)), ...,
       conf.level = NULL,
       conf.args = list(exact = NULL, simulate.p.value = FALSE,
                        B = 2000, col = NA, border = NULL))

x          qqplot
y
xlabylabmain  xlabylabdatax = TRUE
plot.it
datax
distribution
probs
qtype        typequantile
ylim...
conf.level   NULL
conf.args    exactNULLsimulate.p.valueBcolborderNANULL
```

```
qqnormqqplot

x

y          yNAconf.levelqqplotlwrupr
```

```
ppointsqqnorm
```

```
require(graphics)

y <- rt(200, df = 5)
qqnorm(y); qqline(y, col = 2)
qqplot(y, rt(300, df = 5))

qqnorm(precip, ylab = "Precipitation [in/yr] for 70 US cities")

## "QQ-Chisquare" : -----
y <- rchisq(500, df = 3)
## Q-Q plot for Chi^2 data against true theoretical distribution:
qqplot(qchisq(ppoints(500), df = 3), y,
       main = expression("Q-Q plot for" ~~ {chi^2}[nu == 3]))
qqline(y, distribution = function(p) qchisq(p, df = 3),
       probs = c(0.1, 0.6), col = 2)
mtext("qqline(*, dist = qchisq(., df=3), prob = c(0.1, 0.6))")
## (Note that the above uses ppoints() with a = 1/2, giving the
## probability points for quantile type 5: so theoretically, using
## qqline(qtype = 5) might be preferable.)

## Figure 1 in Switzer (1976), knee angle data
switzer <- data.frame(
  angle = c(-31, -30, -25, -25, -23, -23, -22, -20, -20, -18,
            -18, -18, -16, -15, -15, -14, -13, -11, -10, - 9,
            - 8, - 7, - 7, - 7, - 6, - 6, - 4, - 4, - 3, - 2,
            - 2, - 1,  1,  1,  4,  5, 11, 12, 16, 34,
            -31, -20, -18, -16, -16, -16, -15, -14, -14, -14,
            -14, -13, -13, -11, -11, -10, - 9, - 9, - 8, - 7,
            - 7, - 6, - 6, -5, - 5, - 5, - 4, - 2, - 2, - 2,
            0,  0,  1,  1,  2,  4,  5,  5,  6, 17),
  sex = gl(2, 40, labels = c("Female", "Male")))

ks.test(angle ~ sex, data = switzer)
d <- with(switzer, split(angle, sex))
with(d, qqplot(Female, Male, pch = 19, xlim = c(-31, 31), ylim = c(-31, 31),
               conf.level = 0.945,
               conf.args = list(col = "lightgrey", exact = TRUE))
)
abline(a = 0, b = 1)
```



```
## agreement with ks.test
set.seed(1)
x <- rnorm(50)
y <- rnorm(50, mean = .5, sd = .95)
ex <- TRUE
### p = 0.112
(pval <- ks.test(x, y, exact = ex)$p.value)
## 88.8% confidence band with bisecting line
## touching the lower bound
qqplot(x, y, pch = 19, conf.level = 1 - pval,
        conf.args = list(exact = ex, col = "lightgrey"))
abline(a = 0, b = 1)
```

quade.test

```
quade.test(y, ...)

## Default S3 method:
quade.test(y, groups, blocks, ...)

## S3 method for class 'formula'
quade.test(formula, data, subset, na.action, ...)

y
groups      yy
blocks      yy
formula     a ~ b | cabc
data        model.frameformulaenvironment(formula)
subset
na.action   NAgetOption("na.action")
...

quade.testygroupsblocks
blocksygroups
ygroupsblocksNAgroupsblocksyNA

"htest"
statistic
parameter
p.value
method      "Quade test"
data.name
```

friedman.test

```
## Conover (1999, p. 375f):
## Numbers of five brands of a new hand lotion sold in seven stores
## during one week.
y <- matrix(c( 5,  4,  7, 10, 12,
               1,  3,  1,  0,  2,
              16, 12, 22, 22, 35,
               5,  4,  3,  5,  4,
              10,  9,  7, 13, 10,
              19, 18, 28, 37, 58,
              10,  7,  6,  8,  7),
            nrow = 7, byrow = TRUE,
            dimnames =
              list(Store = as.character(1:7),
                   Brand = LETTERS[1:5]))
y
(qTst <- quade.test(y))

## Show equivalence of different versions of test :
utils::str(dy <- as.data.frame(as.table(y)))
qT. <- quade.test(Freq ~ Brand|Store, data = dy)
qT.$data.name <- qTst$data.name
stopifnot(all.equal(qTst, qT., tolerance = 1e-15))
dys <- dy[order(dy[, "Freq"]),]
qTs <- quade.test(Freq ~ Brand|Store, data = dys)
qTs$data.name <- qTst$data.name
stopifnot(all.equal(qTst, qTs, tolerance = 1e-15))
```

quantile

quantile

quantile(x, ...)

```
## Default S3 method:
quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
         names = TRUE, type = 7, digits = 7,
         fuzz = if(type == 7L) 0 else 4 * .Machine$double.eps,
         ...)
```

```

x          NNaNna.rmTRUE
probs      [0,1]2e-14
na.rm      NNaNx
names      namesFALSEprobs
type
digits     namesmax(2, getOption("digits"))
fuzz       j <- floor(np + m)npn * probs
...

length(probs)names = TRUEnames
NNaNNprobs
sortsortquantile
"POSIXt""Date"
fuzz := 4 * .Machine$double.eps type = 4,5,6, 8,9 type0 type = 7

```

```

quantilexprobstype
i

```

$$Q_i(p) = (1 - \gamma)x_j + \gamma x_{j+1}$$

$$1 \leq i \leq 9 \frac{j-m}{n} \leq p < \frac{j-m+1}{n} x_j j n \gamma j = \lfloor np + m \rfloor g = np + m - jm$$

$$Q_i(p)pm = 0i = 1i = 2m = -1/2i = 3$$

$$\gamma = 0g = 0$$

$$\gamma = 0.5g = 0$$

$$\gamma = 0g = 0j$$

$$Q_i(p)p\gamma = gm(p_k, x_k)x_k k p_k$$

$$m = 0p_k = \frac{k}{n}$$

$$m = 1/2p_k = \frac{k-0.5}{n}$$

$$m = pp_k = \frac{k}{n+1}p_k = [F(x_k)]$$

$$m = 1 - pp_k = \frac{k-1}{n-1}p_k = [F(x_k)]$$

$$m = (p+1)/3p_k = \frac{k-1/3}{n+1/3}p_k \approx [F(x_k)]x$$

$$m = p/4 + 3/8p_k = \frac{k-3/8}{n+1/4}x$$

```

fuzz

```

<https://blogs.sas.com/content/iml/2017/05/24/definitions-sample-quantiles.html>
https://en.wikipedia.org/wiki/Quantile#Estimating_quantiles_from_a_sample

[ecdfquantileboxplot.statsfivenum](#)

```
quantile(x <- rnorm(1001)) # Extremes & Quartiles by default
quantile(x, probs = c(0.1, 0.5, 1, 2, 5, 10, 50, NA)/100)

### Compare different types
quantAll <- function(x, prob, ...)
  t(vapply(1:9, function(typ) quantile(x, probs = prob, type = typ, ...),
        quantile(x, prob, type=1, ...)))
p <- c(0.1, 0.5, 1, 2, 5, 10, 50)/100
signif(quantAll(x, p), 4)

## 0% and 100% are equal to min(), max() for all types:
stopifnot(t(quantAll(x, prob=0:1)) == range(x))

## for complex numbers:
z <- complex(real = x, imaginary = -10*x)
signif(quantAll(z, p), 4)
```

r2dtable

r2dtable(n, r, c)

n	
r	integer
c	integer

n

```

## Fisher's Tea Drinker data.
TeaTasting <-
matrix(c(3, 1, 1, 3),
       nrow = 2,
       dimnames = list(Guess = c("Milk", "Tea"),
                        Truth = c("Milk", "Tea")))
## Simulate permutation test for independence based on the maximum
## Pearson residuals (rather than their sum).
rowTotals <- rowSums(TeaTasting)
colTotals <- colSums(TeaTasting)
nOfCases <- sum(rowTotals)
expected <- outer(rowTotals, colTotals) / nOfCases
maxSqResid <- function(x) max((x - expected) ^ 2 / expected)
simMaxSqResid <-
  sapply(r2dtable(1000, rowTotals, colTotals), maxSqResid)
sum(simMaxSqResid >= maxSqResid(TeaTasting)) / 1000
## Fisher's exact test gives p = 0.4857 ...

```

read.ftable

ftable

```

read.ftable(file, sep = "", quote = "\"",
            row.var.names, col.vars, skip = 0)

write.ftable(x, file = "", quote = TRUE, append = FALSE,
            digits = getOption("digits"), sep = " ", ...)

## S3 method for class 'ftable'
format(x, quote = TRUE, digits = getOption("digits"),
      method = c("non.compact", "row.compact", "col.compact", "compact"),
      lsep = " | ",
      justify = c("left", "right"),
      ...)

## S3 method for class 'ftable'
print(x, digits = getOption("digits"), ...)

```

```

file          connection""
sep
quote          read.ftablequote=""write.table
row.var.names
col.vars
skip
x              "ftable"

```

```

append      TRUEfile"|cmd"write.ftableFALSEfile
digits      x
method      "ftable"write.ftable()print
            "ftable"

```

```

            lsep
lsep         method = "compact"
justify      characterformat(..)
...          write()print()methodformat()

```

```

read.ftableskiprow.var.namescol.vars
read.tablextabs
write.ftablemethod

```

ftable

```

## Agresti (1990), page 157, Table 5.8.
## Not in ftable standard format, but o.k.
file <- tempfile()
cat("          Intercourse\n",
    "Race Gender      Yes No\n",
    "White Male       43 134\n",
    "      Female      26 149\n",
    "Black Male        29  23\n",
    "      Female      22  36\n",
    file = file)
file.show(file)
ft1 <- read.ftable(file)
ft1
unlink(file)

## Agresti (1990), page 297, Table 8.16.
## Almost o.k., but misses the name of the row variable.
file <- tempfile()
cat("          \nTonsil Size\n",
    "          \nNot Enl.\n" \nEnl.\n" \nGreatly Enl.\n",
    "Noncarriers      497      560      269\n",
    "Carriers          19       29       24\n",
    file = file)
file.show(file)
ft <- read.ftable(file, skip = 2,
                  row.var.names = "Status",
                  col.vars = list("Tonsil Size" =
                                c("Not Enl.", "Enl.", "Greatly Enl.")))

```

```

ft
unlink(file)

ft22 <- ftable(Titanic, row.vars = 2:1, col.vars = 4:3)
write.ftable(ft22, quote = FALSE) # is the same as
print(ft22)#method="non.compact" is default
print(ft22, method="row.compact")
print(ft22, method="col.compact")
print(ft22, method="compact")

## using 'justify' and 'quote' :
format(ftable(wool + tension ~ breaks, warpbreaks),
       justify = "none", quote = FALSE)

```

rect.hclust

```

rect.hclust(tree, k = NULL, which = NULL, x = NULL, h = NULL,
            border = 2, cluster = NULL)

```

tree	hclust
kh	kh
whichx	whichxwhich = 1:k
border	
cluster	cutree(hclust.obj, k = k)

[hclustidentify.hclust](#)

```

require(graphics)

hca <- hclust(dist(USArrests))
plot(hca)
rect.hclust(hca, k = 3, border = "red")
x <- rect.hclust(hca, h = 50, which = c(2,7), border = 3:4)
x

```

relevel

refcontr.treatment

relevel(x, ref, ...)

x
ref
...

`reorder()``factor`(x, levels = levels(x)[....])

x

`factorcontr.treatmentlevelsreorder`

```
warpbreaks$tension <- relevel(warpbreaks$tension, ref = "M")  
summary(lm(breaks ~ wool + tension, data = warpbreaks))
```

reorder.default

reorder"default"

reorder(x, ...)

```
## Default S3 method:  
reorder(x, X, FUN = mean, ...,  
        order = is.ordered(x), decreasing = FALSE)
```

x	<code>factor</code> x
X	xx
FUN	<code>function</code> Xx
...	FUN
order	
decreasing	


```
relevel()factor(x, levels = levels(x)[...])
```

```
orderFUNxFUN  
FUNx"scores"
```

```
<deepayan.sarkar@r-project.org>
```

```
reorder.dendrogramlevelsrelevel
```

```
require(graphics)
```

```
bymedian <- with(InsectSprays, reorder(spray, count, median))  
boxplot(count ~ bymedian, data = InsectSprays,  
         xlab = "Type of spray", ylab = "Insect count",  
         main = "InsectSprays data", varwidth = TRUE,  
         col = "lightgray")
```

```
bymedianR <- with(InsectSprays, reorder(spray, count, median, decreasing=TRUE))  
stopifnot(exprs = {  
  identical(attr(bymedian, "scores") -> sc,  
            attr(bymedianR, "scores"))  
  identical(nms <- names(sc), LETTERS[1:6])  
  identical(levels(bymedian ), nms[isc <- order(sc)])  
  identical(levels(bymedianR), nms[rev(isc)])  
})
```

```
reorder.dendrogram
```

```
reorder
```

```
## S3 method for class 'dendrogram'  
reorder(x, wts, agglo.FUN = sum, ...)
```

```
x  
wts  
agglo.FUN  
...
```

```
 $wts f(w_j) fagglo.FUN w_j j$ 
```

value

[reorder](#)

[rev.dendrogramheatmapcophenetic](#)

```
require(graphics)
```

```
set.seed(123)
x <- rnorm(10)
hc <- hclust(dist(x))
dd <- as.dendrogram(hc)
dd.reorder <- reorder(dd, 10:1)
plot(dd, main = "random dendrogram 'dd'")

op <- par(mfcol = 1:2)
plot(dd.reorder, main = "reorder(dd, 10:1)")
plot(reorder(dd, 10:1, agglo.FUN = mean), main = "reorder(dd, 10:1, mean)")
par(op)
```

replications

```
replications(formula, data = NULL, na.action)
```

formula

data formula

na.action na.actiondatana.actionna.fail

```
formuladataformuladata~ .
```

```
!is.list(replications(formula,data))
```

model.tables

```
## From Venables and Ripley (2002) p.165.
N <- c(0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,0,1,0,1,0,1,1,0,0)
P <- c(1,1,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,0,0,1,0,1,1,0)
K <- c(1,0,0,1,0,1,1,0,0,1,0,1,0,1,1,0,0,0,1,1,1,0,1,0)
yield <- c(49.5,62.8,46.8,57.0,59.8,58.5,55.5,56.0,62.8,55.8,69.5,
55.0, 62.0,48.8,45.5,44.2,52.0,51.5,49.8,48.8,57.2,59.0,53.2,56.0)

npk <- data.frame(block = gl(6,4), N = factor(N), P = factor(P),
                  K = factor(K), yield = yield)
replications(~ . - yield, npk)
```

reshape

```
reshape(data, varying = NULL, v.names = NULL, timevar = "time",
        idvar = "id", ids = 1:NROW(data),
        times = seq_along(varying[[1]]),
        drop = NULL, direction, new.row.names = NULL,
        sep = ".",
        split = if (sep == "") {
          list(regexp = "[A-Za-z][0-9]", include = TRUE)
        } else {
          list(regexp = sep, include = FALSE, fixed = TRUE)}
        )

### Typical usage for converting from long to wide format:

# reshape(data, direction = "wide",
#         idvar = "___", timevar = "___", # mandatory
#         v.names = c(___), # time-varying variables
#         varying = list(___)) # auto-generated if missing

### Typical usage for converting from wide to long format:

### If names of wide-format variables are in a 'nice' format

# reshape(data, direction = "long",
#         varying = c(___), # vector
#         sep) # to help guess 'v.names' and 'times'
```

```
# reshape(data, direction = "long",
#         varying = ___, # list / matrix / vector (use with care)
#         v.names = ___, # vector of variable names in long format
#         timevar, times, # name / values of constructed time variable
#         idvar, ids)     # name / values of constructed id variable
```

reshape()

```
stackpermrelistunlistxtabsas.data.frame.table
vignette("reshape")
```

```

summary(Indometh) # data in long format

## long to wide (direction = "wide") requires idvar and timevar at a minimum
reshape(Indometh, direction = "wide", idvar = "Subject", timevar = "time")

## can also explicitly specify name of combined variable
wide <- reshape(Indometh, direction = "wide", idvar = "Subject",
               timevar = "time", v.names = "conc", sep = "_")
wide

## reverse transformation
reshape(wide, direction = "long")
reshape(wide, idvar = "Subject", varying = list(2:12),
       v.names = "conc", direction = "long")

## times need not be numeric
df <- data.frame(id = rep(1:4, rep(2,4)),
               visit = rep(c("Before","After"), 4),
               x = rnorm(4), y = runif(4))
df
reshape(df, timevar = "visit", idvar = "id", direction = "wide")
## warns that y is really varying
reshape(df, timevar = "visit", idvar = "id", direction = "wide", v.names = "x")

## unbalanced 'long' data leads to NA fill in 'wide' form
df2 <- df[1:7, ]
df2
reshape(df2, timevar = "visit", idvar = "id", direction = "wide")

## Alternative regular expressions for guessing names
df3 <- data.frame(id = 1:4, age = c(40,50,60,50), dose1 = c(1,2,1,2),
               dose2 = c(2,1,2,1), dose4 = c(3,3,3,3))
reshape(df3, direction = "long", varying = 3:5, sep = "")

## an example that isn't longitudinal data
state.x77 <- as.data.frame(state.x77)
long <- reshape(state.x77, idvar = "state", ids = row.names(state.x77),
               times = names(state.x77), timevar = "Characteristic",
               varying = list(names(state.x77)), direction = "long")

reshape(long, direction = "wide")

reshape(long, direction = "wide", new.row.names = unique(long$state))

## multiple id variables
df3 <- data.frame(school = rep(1:3, each = 4), class = rep(9:10, 6),
               time = rep(c(1,1,2,2), 3), score = rnorm(12))
wide <- reshape(df3, idvar = c("school", "class"), direction = "wide")
wide
## transform back
reshape(wide)

```

residuals

residuals
residresiduals
residualsresidualsresid
[naresidnlssmooth.spline](#)

residuals(object, ...)
resid(object, ...)

object
...

object

[coefficientsfitted.valuesglm](#)
[rstandardrstudent](#)

runmed

```
runmed(x, k, endrule = c("median", "keep", "constant"),
      algorithm = NULL,
      na.action = c("+Big_alterate", "-Big_alterate", "na.omit", "fail"),
      print.level = 0)

x
k      k <- 1 + 2 * min((n-1)%/% 2, ceiling(0.1*n))k = 3
endrule
      "keep"  k2k2k2 = k %/% 2y[j] = x[j]j ∈ {1,...,k2;n - k2 + 1,...,n}
      "constant" median(y[1:k2])
      "median" smoothEnds
```

```

algorithm      "Turlach""Stuetzle"NULLn = length(x)k"Turlach"
na.action      NANAx
               "+Big_alternate" x±BB2B < M*M* =.Machine $ double.xmax
               (+B, -B, +B, ...)"+"
               "-Big_alternate" "+Big_alternate"-B"-Big..."
               "na.omit" runmed(x[!is.na(x)], k, ..)
               "fail" x

print.level

```

```

y = runmed(x, k)y[j] = median(x[(j-k2):(j+k2)])k = 2*k2+1

```

```

"Turlach"  $O(n \log k)$ n = length(x)

```

```

"Stuetzle"  $O(n \times k)kn$ 

```

```

smoothEnds()x±InfNaNNA

```

```

"Turlach"

```

```

"Stuetzle" libm

```

```

algorithm = "Stuetzle"

```

```

xattrkk

```

```

<maechler@stat.math.ethz.ch>

```

```

smoothEndsrunmed(*, endrule = "median")smooth

```

```

require(graphics)

```

```

utils::example(nhtemp)

```

```

myNHT <- as.vector(nhtemp)

```

```

myNHT[20] <- 2 * nhtemp[20]

```

```

plot(myNHT, type = "b", ylim = c(48, 60), main = "Running Medians Example")

```

```

lines(runmed(myNHT, 7), col = "red")

```

```

## special: multiple y values for one x

```

```

plot(cars, main = "'cars' data and runmed(dist, 3)")

```

```

lines(cars, col = "light gray", type = "c")

```

```

with(cars, lines(speed, runmed(dist, k = 3), col = 2))

```

```

## nice quadratic with a few outliers

```

```

y <- ys <- (-20:20)^2
y [c(1,10,21,41)] <- c(150, 30, 400, 450)
all(y == runmed(y, 1)) # 1-neighbourhood <==> interpolation
plot(y) ## lines(y, lwd = .1, col = "light gray")
lines(lowess(seq(y), y, f = 0.3), col = "brown")
lines(runmed(y, 7), lwd = 2, col = "blue")
lines(runmed(y, 11), lwd = 2, col = "red")

## Lowess is not robust
y <- ys ; y[21] <- 6666 ; x <- seq(y)
col <- c("black", "brown", "blue")
plot(y, col = col[1])
lines(lowess(x, y, f = 0.3), col = col[2])
lines(runmed(y, 7), lwd = 2, col = col[3])
legend(length(y), max(y), c("data", "lowess(y, f = 0.3)", "runmed(y, 7)"),
      xjust = 1, col = col, lty = c(0, 1, 1), pch = c(1, NA, NA))

## An example with initial NA's - used to fail badly (notably for "Turlach"):
x15 <- c(rep(NA, 4), c(9, 9, 4, 22, 6, 1, 7, 5, 2, 8, 3))
rS15 <- cbind(Sk.3 = runmed(x15, k = 3, algorithm="S"),
             Sk.7 = runmed(x15, k = 7, algorithm="S"),
             Sk.11 = runmed(x15, k = 11, algorithm="S"))
rT15 <- cbind(Tk.3 = runmed(x15, k = 3, algorithm="T", print.level=1),
             Tk.7 = runmed(x15, k = 7, algorithm="T", print.level=1),
             Tk.9 = runmed(x15, k = 9, algorithm="T", print.level=1),
             Tk.11 = runmed(x15, k = 11, algorithm="T", print.level=1))
cbind(x15, rS15, rT15) # result for k=11 maybe a bit surprising ..
Tv <- rT15[-(1:3),]
stopifnot(3 <= Tv, Tv <= 9, 5 <= Tv[1:10,])
matplot(y = cbind(x15, rT15), type = "b", ylim = c(1,9), pch=1:5, xlab = NA,
      main = "runmed(x15, k, algo = \"Turlach\")")
mtext(paste("x15 <- ", deparse(x15)))
points(x15, cex=2)
legend("bottomleft", legend=c("data", paste("k = ", c(3,7,9,11))),
      bty="n", col=1:5, lty=1:5, pch=1:5)

```

rWishart

$n\text{Sigmadf}W_p(\Sigma, m), m = \text{df}, \Sigma = \text{Sigma}$

$\text{rWishart}(n, \text{df}, \text{Sigma})$

n

df

$\text{Sigma} \quad p \times p$

$$X_1, \dots, X_m, X_i \in \mathbf{R}^p, m \Sigma M = X' X W_p(\Sigma, m)$$

M

$$E[M] = m \times \Sigma.$$

$$\text{Sigma}p = 1\chi^2\text{df}W_1(\sigma^2, m) = \sigma^2\chi_m^2$$

$$\text{Var}(M_{ij}) = m(\Sigma_{ij}^2 + \Sigma_{ii}\Sigma_{jj}).$$

$$\text{array}R_{p \times p \times n}[\,,i]W_p(\Sigma, m), m = \text{df}, \Sigma = \text{Sigma}$$

`covrnormrchisq`

```
## Artificial
S <- toeplitz((10:1)/10)
set.seed(11)
R <- rWishart(1000, 20, S)
dim(R) # 10 10 1000
mR <- apply(R, 1:2, mean) # ~= E[ Wish(S, 20) ] = 20 * S
stopifnot(all.equal(mR, 20*S, tolerance = .009))

## See Details, the variance is
Va <- 20*(S^2 + tcrossprod(diag(S)))
vR <- apply(R, 1:2, var)
stopifnot(all.equal(vR, Va, tolerance = 1/16))
```

<code>scatter.smooth</code>	<code>loess</code>
-----------------------------	--------------------

`loess`

```
scatter.smooth(x, y = NULL, span = 2/3, degree = 1,
  family = c("symmetric", "gaussian"),
  xlab = NULL, ylab = NULL,
  ylim = range(y, pred$y, na.rm = TRUE),
  evaluation = 50, ..., lpars = list())

loess.smooth(x, y, span = 2/3, degree = 1,
  family = c("symmetric", "gaussian"), evaluation = 50, ...)
```

xy	xyxy.coords
span	loess
degree	
family	"gaussian"family = "symmetric"
xlab	
ylab	
ylim	
evaluation	
...	scatter.smooth()plot()loess.smoothloess.control
lpars	listlines()

loess.smoothloessevaluationx

scatter.smooth

loess.smoothxy

loesssmoothScatter

require(graphics)

```
with(cars, scatter.smooth(speed, dist))
## or with dotted thick smoothed line results :
with(cars, scatter.smooth(speed, dist, lpars =
  list(col = "red", lwd = 3, lty = 3)))
```

screepplot

screepplot.defaultplot"princomp""prcomp"

```
screepplot(x, ...)
## Default S3 method:
screepplot(x, npcs = min(10, length(x$sdev)),
  type = c("barplot", "lines"),
  main = deparse1(substitute(x)), ...)
```

x sdevprincomp()prcomp()

npcs

type

main...

princomp
princomp

```
require(graphics)

## The variances of the variables in the
## USArrests data vary by orders of magnitude, so scaling is appropriate
(pc.cr <- princomp(USArrests, cor = TRUE)) # inappropriate
screeplot(pc.cr)

fit <- princomp(covmat = Harman74.cor)
screeplot(fit)
screeplot(fit, npcs = 24, type = "lines")
```

sd

xna.rmTRUE

sd(x, na.rm = FALSE)

x factoras.double(x)
na.rm

var $n - 1$

NA

varmad

sd(1:2) ^ 2

se.contrast

aov

```
se.contrast(object, ...)
## S3 method for class 'aov'
se.contrast(object, contrast.obj,
             coef = contr.helmert(ncol(contrast))[, 1],
             data = NULL, ...)
```

object	aov
contrast.obj	
coef	contrast.obj
data	contrast.obj
...	

se.contrast

[aov](#)

coef[contrasts](#)

[contrastsmodel.tables](#)

```
## From Venables and Ripley (2002) p.165.
N <- c(0,1,0,1,1,1,0,0,0,1,1,0,1,1,0,0,1,0,1,0,1,1,0,0)
P <- c(1,1,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,0,0,1,0,1,1,0)
K <- c(1,0,0,1,0,1,1,0,0,1,0,1,0,1,1,0,0,0,1,1,1,0,1,0)
yield <- c(49.5,62.8,46.8,57.0,59.8,58.5,55.5,56.0,62.8,55.8,69.5,
55.0, 62.0,48.8,45.5,44.2,52.0,51.5,49.8,48.8,57.2,59.0,53.2,56.0)

npk <- data.frame(block = gl(6,4), N = factor(N), P = factor(P),
                  K = factor(K), yield = yield)
## Set suitable contrasts.
options(contrasts = c("contr.helmert", "contr.poly"))
npk.aov1 <- aov(yield ~ block + N + K, data = npk)
se.contrast(npk.aov1, list(N == "0", N == "1"), data = npk)
# or via a matrix
cont <- matrix(c(-1,1), 2, 1, dimnames = list(NULL, "N"))
se.contrast(npk.aov1, cont[N, , drop = FALSE]/12, data = npk)

## test a multi-stratum model
```

selfStart

```
nlsinitialstartnls()
```

```
selfStart(model, initial, parameters, template)
```

```
model          formula~ expression
initial        mCalldataLHS...modelmCallnlsmode...nls()controltraceinitial()
               nls()
parameters     modelnamevecderiv
template       function.argderivmodelmodel
```

```
nls()getInitialinitial
```

```
function"selfStart"formuladerivmodelinitialinitial
```

```
nlsgetInitial
```

```
"selfStart"SSasymSSasymOffSSasymOrigSSbiexpSSfolSSfplSSgompertzSSlogis
SSmicmenSSweibull
```

```
nlsList
```

```
## self-starting logistic model
```

```
## The "initializer" (finds initial values for parameters from data):
initLogis <- function(mCall, data, LHS, ...) {
  xy <- sortedXyData(mCall[["x"]], LHS, data)
  if(nrow(xy) < 4)
    stop("too few distinct input values to fit a logistic model")
  z <- xy[["y"]]
  ## transform to proportion, i.e. in (0,1) :
  rng <- range(z); dz <- diff(rng)
  z <- (z - rng[1L] + 0.05 * dz)/(1.1 * dz)
  xy[["z"]] <- log(z/(1 - z)) # logit transformation
  aux <- coef(lm(x ~ z, xy))
  pars <- coef(nls(y ~ 1/(1 + exp((xmid - x)/scal)),
    data = xy,
    start = list(xmid = aux[[1L]], scal = aux[[2L]]),
    algorithm = "plinear", ...)
  setNames(pars [c(".lin", "xmid", "scal")],
    mCall[c("Asym", "xmid", "scal")])
}
```

```
mySSlogis <- selfStart(~ Asym/(1 + exp((xmid - x)/scal)),
  initial = initLogis,
  parameters = c("Asym", "xmid", "scal"))
```

```

getInitial(weight ~ mySSlogis(Time, Asym, xmid, scal),
           data = subset(ChickWeight, Chick == 1))

# 'first.order.log.model' is a function object defining a first order
# compartment model
# 'first.order.log.initial' is a function object which calculates initial
# values for the parameters in 'first.order.log.model'
#
# self-starting first order compartment model
## Not run:
SSfol <- selfStart(first.order.log.model, first.order.log.initial)

## End(Not run)

## Explore the self-starting models already available in R's "stats":
pos.st <- which("package:stats" == search())
mSS <- apropos("^SS..", where = TRUE, ignore.case = FALSE)
(mSS <- unname(mSS[names(mSS) == pos.st]))
fSS <- sapply(mSS, get, pos = pos.st, mode = "function")
all(sapply(fSS, inherits, "selfStart")) # -> TRUE

## Show the argument list of each self-starting function:
str(fSS, give.attr = FALSE)

```

setNames

```
setNames(object = nm, nm)
```

```

object      names
nm

```

```
object
```

unname

```

setNames( 1:3, c("foo", "bar", "baz") )
# this is just a short form of
tmp <- 1:3
names(tmp) <- c("foo", "bar", "baz")
tmp

```

```

## special case of character vector, using default
setNames(nm = c("First", "2nd"))

```

shapiro.test

shapiro.test(x)

x

"htest"

statistic

p.value p.value < 0.1

method "Shapiro-Wilk normality test"

data.name

$n = 34 \leq n \leq 11n \geq 12$

W

W

W

qqnorm

```
shapiro.test(rnorm(100, mean = 5, sd = 3))
shapiro.test(runif(100, min = 2, max = 4))
```

sigma

summary.lm()

σ sigma(.) $\hat{\sigma}$

sigma(object, ...)

Default S3 method:

sigma(object, use.fallback = TRUE, ...)


```

object      lm
use.fallback nobs
...         deviance(*, ...)

```

```
"glm"
```

```

sigma.default <- function (object, use.fallback = TRUE, ...)
  sqrt( deviance(object, ...) / (NN - PP) )

```

```

NN <- nobs(object, use.fallback = use.fallback)
PP <- sum(!is.na(coef(object)))
length(coef(object))

```

$$\hat{\sigma} \sigma \sqrt{\sigma^2}$$

```

"glm"sigma.glmsummary.glm
"mlm"Y

```

```
deviancenobsvcovsummary.glm
```

```

## -- lm() -----
lm1 <- lm(Fertility ~ . , data = swiss)
sigma(lm1) # ~= 7.165 = "Residual standard error" printed from summary(lm1)
stopifnot(all.equal(sigma(lm1), summary(lm1)$sigma, tolerance=1e-15))

## -- nls() -----
DNase1 <- subset(DNase, Run == 1)
fm.DN1 <- nls(density ~ SSlogis(log(conc), Asym, xmid, scal), DNase1)
sigma(fm.DN1) # ~= 0.01919 as from summary(..)
stopifnot(all.equal(sigma(fm.DN1), summary(fm.DN1)$sigma, tolerance=1e-15))

## -- glm() -----
## -- a) Binomial -- Example from MASS
ldose <- rep(0:5, 2)
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
sex <- factor(rep(c("M", "F"), c(6, 6)))
SF <- cbind(numdead, numalive = 20-numdead)
sigma(budworm.lg <- glm(SF ~ sex*ldose, family = binomial))

## -- b) Poisson -- from ?glm :
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)

```

```

treatment <- gl(3,3)
sigma(glm.D93 <- glm(counts ~ outcome + treatment, family = poisson()))
## equal to
sqrt(summary(glm.D93)$dispersion) # == 1
## and the *Quasi*poisson's dispersion
sigma(glm.qD93 <- update(glm.D93, family = quasipoisson()))
sigma (glm.qD93)^2 # 1.2933 equal to
summary(glm.qD93)$dispersion # == 1.2933

## -- Multivariate lm() "mlm" -----
utils::example("SSD", echo=FALSE)
sigma(mlmfit) # is the same as {but more efficient than}
sqrt(diag(estVar(mlmfit)))

```

SignRank

n

```

dsignrank(x, n, log = FALSE)
psignrank(q, n, lower.tail = TRUE, log.p = FALSE)
qsignrank(p, n, lower.tail = TRUE, log.p = FALSE)
rsignrank(nn, n)

```

xq

p

nn length(nn) > 1

n

loglog.p

lower.tail $P[X \leq x]P[X > x]$

$xn x[i]x[i]0n(n+1)/2n(n+1)/4n(n+1)(2n+1)/24$

dsignrankpsignrankqsignrankrsignrank

nnrsignrank

nn

[wilcox.test](#)

[dwilcox](#)

```
require(graphics)

par(mfrow = c(2,2))
for(n in c(4:5,10,40)) {
  x <- seq(0, n*(n+1)/2, length.out = 501)
  plot(x, dsignrank(x, n = n), type = "l",
       main = paste0("dsignrank(x, n = ", n, ")"))
}
```

simulate

```
simulate(object, nsim = 1, seed = NULL, ...)
```

```
object
```

```
nsim          1
```

```
seed
```

```
"lm"NULLset.seed"seed"NULL.Random.seed"seed"
```

```
...
```

```
"lm"lmglmglm.nb"lm"
```

```
lmglm(family = "gaussian")simulatefamily
```

```
 $w_i w_i$ 
```

```
gamma.shape
```

```
 $IG(\mu_i, \lambda w_i)$ https://en.wikipedia.org/wiki/Inverse\_Gaussian\_distribution $\lambda \mu_i^3 / (\lambda w_i)$ 
```

```
rinvGauss
```

```
nsim
```

```
"lm""seed"seedNULL.Random.seed"kind"as.list(RNGkind())
```

```
RNGfitted.valuesresidualsglm
```

```
simulate.R
```

```

x <- 1:5
mod1 <- lm(c(1:3, 7, 6) ~ x)
S1 <- simulate(mod1, nsim = 4)
## repeat the simulation:
.Random.seed <- attr(S1, "seed")
identical(S1, simulate(mod1, nsim = 4))

S2 <- simulate(mod1, nsim = 200, seed = 101)
rowMeans(S2) # should be about the same as
fitted(mod1)

## repeat identically:
(sseed <- attr(S2, "seed")) # seed; RNGkind as attribute
stopifnot(identical(S2, simulate(mod1, nsim = 200, seed = sseed)))

## To be sure about the proper RNGkind, e.g., after
RNGversion("2.7.0")
## first set the RNG kind, then simulate
do.call(RNGkind, attr(sseed, "kind"))
identical(S2, simulate(mod1, nsim = 200, seed = sseed))

## Binomial GLM examples
yb1 <- matrix(c(4, 4, 5, 7, 8, 6, 6, 5, 3, 2), ncol = 2)
modb1 <- glm(yb1 ~ x, family = binomial)
S3 <- simulate(modb1, nsim = 4)
# each column of S3 is a two-column matrix.

x2 <- sort(runif(100))
yb2 <- rbinom(100, prob = plogis(2*(x2-1)), size = 1)
yb2 <- factor(1 + yb2, labels = c("failure", "success"))
modb2 <- glm(yb2 ~ x2, family = binomial)
S4 <- simulate(modb2, nsim = 4)
# each column of S4 is a factor

```

Smirnov

```

psmirnov(q, sizes, z = NULL,
         alternative = c("two.sided", "less", "greater"),
         exact = TRUE, simulate = FALSE, B = 2000,
         lower.tail = TRUE, log.p = FALSE)
qsmirnov(p, sizes, z = NULL,
         alternative = c("two.sided", "less", "greater"),
         exact = TRUE, simulate = FALSE, B = 2000)
rsmirnov(n, sizes, z = NULL,
         alternative = c("two.sided", "less", "greater"))

```

q
p
sizes
z
alternative "two.sided""less""greater"
exact NULLz
simulate
B
lower.tail TRUE $P[D < q]P[D \geq q]$
log.p TRUE
n

$xy n_x n_y F_{x,n_x} F_{y,n_y}$

$$D = \sup_c |F_{x,n_x}(c) - F_{y,n_y}(c)|$$
$$D^+ = \sup_c (F_{x,n_x}(c) - F_{y,n_y}(c))$$

"greater"

$$D^- = \sup_c (F_{y,n_y}(c) - F_{x,n_x}(c))$$

"less"

xy [ks.test](#)

$FF Fz xy n_x n_y z$

psmirnovqsmirnovrsmirnov

[ks.test](#)

smooth

smooth(x, kind = c("3RS3R", "3RSS", "3RSR", "3R", "3", "S"),
twiceit = FALSE, endrule = c("Tukey", "copy"), do.ends = FALSE)

x
kind "3RS3R"
twiceit $S(y)S(y) + S(y - S(y))$
endrule "Tukey""copy"
do.ends kind = "S"

```

3median
3R3
S
3RS3R3RS3R3RSS3RSR3R(S and 3)

```

```

"tukeysmooth"printsummary

```

```

smoothendrule
kind = "3RSR"
smooth(*)"3RS*"

```

```

runmedlowessloesssupsmusmooth.spline

```

```

require(graphics)

## see also    demo(smooth) !

x1 <- c(4, 1, 3, 6, 6, 4, 1, 6, 2, 4, 2) # very artificial
(x3R <- smooth(x1, "3R")) # 2 iterations of "3"
smooth(x3R, kind = "S")

sm.3RS <- function(x, ...)
  smooth(smooth(x, "3R", ...), "S", ...)

y <- c(1, 1, 19:1)
plot(y, main = "misbehaviour of \"3RSR\"", col.main = 3)
lines(sm.3RS(y))
lines(smooth(y))
lines(smooth(y, "3RSR"), col = 3, lwd = 2) # the horror

x <- c(8:10, 10, 0, 0, 9, 9)
plot(x, main = "breakdown of 3R and S and hence 3RSS")
matlines(cbind(smooth(x, "3R"), smooth(x, "S"), smooth(x, "3RSS"), smooth(x)))

presidents[is.na(presidents)] <- 0 # silly
summary(sm3 <- smooth(presidents, "3R"))
summary(sm2 <- smooth(presidents, "3RSS"))
summary(sm <- smooth(presidents))

all.equal(c(sm2), c(smooth(smooth(sm3, "S"), "S"))) # 3RSS === 3R S S
all.equal(c(sm), c(smooth(smooth(sm3, "S"), "3R"))) # 3RS3R === 3R S 3R

plot(presidents, main = "smooth(presidents0, *) : 3R and default 3RS3R")
lines(sm3, col = 3, lwd = 1.5)
lines(sm, col = 2, lwd = 1.25)

```

smooth.spline

```
smooth.spline(x, y = NULL, w = NULL, df, spar = NULL, lambda = NULL, cv = FALSE,
  all.knots = FALSE, nknots = .nknots.smspl,
  keep.data = TRUE, df.offset = 0, penalty = 1,
  control.spar = list(), tol = 1e-6 * IQR(x), keep.stuff = FALSE)
```

```
.nknots.smspl(n)
```

x	
y	yNULLxx
w	x
df	(1,n_x]n_x
spar	(0,1]sparλsparlambdaspars
lambda	λspar
cv	TRUEFALSEspardfcv.critNA
all.knots	TRUExFALSEx[]x[j]nknots1:nknknots numeric[0,1]ans \$ fit\$knots
nknots	functionall.knots = FALSEnknots(nx).nknots.smspl() $n_x > 49n_x \times$
keep.data	TRUE
df.offset	df.offset
penalty	
control.spar	sparNULL spar spar
	spar[low,high]
tol	xtol
keep.stuff	logicalX
n	.nknots.smsplxn_x

```

xy
xtolyw
lambdaspar $\lambda s = spar\lambda = r \cdot 256^{3s-1}r = tr(X'WX)/tr(\Sigma)\Sigma\Sigma_{ij} = \int B_i''(t)B_j''(t)dtXX_{ij} = B_j(x_i)WnB_k(.)k$ 
 $f_i = f(x_i)f = XccL = (y - f)'W(y - f) + \lambda c'\Sigma cc(X'WX + \lambda\Sigma)c = X'Wy$ 
sparlambdaNULLdfdfcv $\lambda$ 
spars $s = s0 + 0.0601 \cdot \log \lambda$ 
sparsparlowhighlambdasparlambda
xcv = TRUE

```

```
"smooth.spline"
```

```

x          x
y          x
w          x
yin        y
tol        tol x
data       keep.data = TRUElistxyw( $x_i, y_i, w_i$ ),  $i = 1, \dots, ndata$ $xx
n
lev        cvNA
cv         cvFALSETRUENA
cv.crit    cvprint()critnx < nnx= $n_x$ 
pen.crit   sum(.$w * residuals())^2)
crit       .Fortransslvrgdf3 + ( $tr(S_\lambda) - df$ )^23+
df         df
spar       sparc(lambda = *)NA
ratio      sparNA $r$ 
lambda      $\lambda$ spar
iparms     ..$ipars["iter"]
auxMat     keep.stuff
fit        predict.smooth.spline
          [0,1]minrange

          x

call

method(class = "smooth.spline")hatvalues()lev

```



```
xnx =  $n_x$ tol
```

```
nx <- length(x) - sum(duplicated( round((x - mean(x)) / tol) ))
```

```
all.knots = FALSE  
enknots = .nknots.smspl $O(n_x^{0.2})n_xn_x > 49O(n_k) + O(n)n_k$   
xdf
```

```
spar/lambda
```

```
GAMFIThttps://lib.stat.cmu.edu/general/gamfitsmooth.spline
```

```
predict.smooth.spline
```

```
require(graphics)  
plot(dist ~ speed, data = cars, main = "data(cars) & smoothing splines")  
cars.spl <- with(cars, smooth.spline(speed, dist))  
cars.spl  
## This example has duplicate points, so avoid cv = TRUE  
  
lines(cars.spl, col = "blue")  
ss10 <- smooth.spline(cars[, "speed"], cars[, "dist"], df = 10)  
lines(ss10, lty = 2, col = "red")  
legend(5, 120, c(paste("default [C.V.] => df =", round(cars.spl$df, 1)),  
                "s( * , df = 10)"), col = c("blue", "red"), lty = 1:2,  
       bg = 'bisque')  
  
## Residual (Tukey Anscombe) plot:  
plot(residuals(cars.spl) ~ fitted(cars.spl))  
abline(h = 0, col = "gray")  
  
## consistency check:  
stopifnot(all.equal(cars$dist,  
                    fitted(cars.spl) + residuals(cars.spl)))  
## The chosen inner knots in original x-scale :  
with(cars.spl$fit, min + range * knot[-c(1:3, nk+1 + 1:3)]) # == unique(cars$speed)  
  
## Visualize the behavior of .nknots.smspl()  
nKnots <- Vectorize(.nknots.smspl) ; c.. <- adjustcolor("gray20", .5)  
curve(nKnots, 1, 250, n = 250)  
abline(0, 1, lty = 2, col = c..); text(90, 90, "y = x", col = c.., adj = -.25)
```

```

abline(h=100,lty=2); abline(v=200, lty=2)

n <- c(1:799, seq(800, 3490, by=10), seq(3500, 10000, by = 50))
plot(n, nKnots(n), type="l", main = "Vectorize(.nknots.smspl) (n)")
abline(0,1, lty=2, col=c..); text(180,180,"y = x", col=c..)
n0 <- c(50, 200, 800, 3200); c0 <- adjustcolor("blue3", .5)
lines(n0, nKnots(n0), type="h", col=c0)
axis(1, at=n0, line=-2, col.ticks=c0, col=NA, col.axis=c0)
axis(4, at=.nknots.smspl(10000), line=-.5, col=c..,col.axis=c.., las=1)

##-- artificial example
y18 <- c(1:3, 5, 4, 7:3, 2*(2:5), rep(10, 4))
xx <- seq(1, length(y18), length.out = 201)
(s2 <- smooth.spline(y18)) # GCV
(s02 <- smooth.spline(y18, spar = 0.2))
(s02. <- smooth.spline(y18, spar = 0.2, cv = NA))
plot(y18, main = deparse(s2$call), col.main = 2)
lines(s2, col = "gray"); lines(predict(s2, xx), col = 2)
lines(predict(s02, xx), col = 3); mtext(deparse(s02$call), col = 3)

## Specifying 'lambda' instead of usual spar :
(s2. <- smooth.spline(y18, lambda = s2$lambda, tol = s2$tol))

## The following shows the problematic behavior of 'spar' searching:
(s2 <- smooth.spline(y18, control =
  list(trace = TRUE, tol = 1e-6, low = -1.5)))
(s2m <- smooth.spline(y18, cv = TRUE, control =
  list(trace = TRUE, tol = 1e-6, low = -1.5)))
## both above do quite similarly (Df = 8.5 +- 0.2)

```

smoothEnds

y

smoothEnds(y, k = 3)

y

k

smoothEndsksm[1] = median(y[1], sm[2], 3*sm[2] - 2*sm[3], na.rm=TRUE)
[NAymedian](#)(*, na.rm=TRUE)

y

```
runmed(*, endrule = "median")smoothEnds()
```

```
require(graphics)
```

```
y <- ys <- (-20:20)^2
y [c(1,10,21,41)] <- c(100, 30, 400, 470)
s7k <- runmed(y, 7, endrule = "keep")
s7. <- runmed(y, 7, endrule = "const")
s7m <- runmed(y, 7)
col3 <- c("midnightblue", "blue", "steelblue")
plot(y, main = "Running Medians -- runmed(*, k=7, endrule = X)")
lines(ys, col = "light gray")
matlines(cbind(s7k, s7., s7m), lwd = 1.5, lty = 1, col = col3)
eRules <- c("keep", "constant", "median")
legend("topleft", paste("endrule", eRules, sep = " = "),
       col = col3, lwd = 1.5, lty = 1, bty = "n")
```

```
stopifnot(identical(s7m, smoothEnds(s7k, 7)))
```

```
## With missing values (for R >= 3.6.1):
yN <- y; yN[c(2,40)] <- NA
rN <- sapply(eRules, function(R) runmed(yN, 7, endrule=R))
matlines(rN, type = "b", pch = 4, lwd = 3, lty=2,
        col = adjustcolor(c("red", "orange4", "orange1"), 0.5))
yN[c(1, 20:21)] <- NA # additionally
rN. <- sapply(eRules, function(R) runmed(yN, 7, endrule=R))
head(rN., 4); tail(rN.) # more NA's too, still not *so* many:
stopifnot(exprs = {
  !anyNA(rN[,2:3])
  identical(which(is.na(rN[, "keep"])), c(2L, 40L))
  identical(which(is.na(rN.), arr.ind=TRUE, useNames=FALSE),
            cbind(c(1:2, 40L), 1L))
  identical(rN.[38:41, "median"], c(289, 289, 397, 470))
})
```

sortedXyData

sortedXyData

sortedXyDatainitialselfStart

sortedXyData(x, y, data)

x	data
y	data
data	xy

sortedXyDataxyxy

[selfStartNLSstClosestXNLSstLfAsymptoteNLSstRtAsymptote](#)

```
DNase.2 <- DNase[ DNase$Run == "2", ]
sortedXyData( expression(log(conc)), expression(density), DNase.2 )
```

spec.ar

x

```
spec.ar(x, n.freq, order = NULL, plot = TRUE, na.action = na.fail,
        method = "yule-walker", ...)
```

x	ar
n.freq	
order	
plot	
na.action	NA
method	methodar
...	plot.spec

"spec"plot

arspectrum

```
require(graphics)

spec.ar(lh)

spec.ar(ldeaths)
spec.ar(ldeaths, method = "burg")

spec.ar(log(lynx))
spec.ar(log(lynx), method = "burg", add = TRUE, col = "purple")
spec.ar(log(lynx), method = "mle", add = TRUE, col = "forest green")
spec.ar(log(lynx), method = "ols", add = TRUE, col = "blue")
```

spec.pgram

```
spec.pgram

spec.pgram(x, spans = NULL, kernel, taper = 0.1,
           pad = 0, fast = TRUE, demean = FALSE, detrend = TRUE,
           plot = TRUE, na.action = na.fail, ...)
```

x	
spans	
kernel	"tskernel"
taper	
pad	pad
fast	TRUE
demean	TRUE
detrend	TRUE
plot	
na.action	NA
...	plot.spec

fastpad

"spec"spectrum

kernel	kernelspans
df	df
bandwidth	
taper	taper
pad	pad
detrend	detrend
demean	demean
plot	

spectrumspec.taperplot.specfft

```
require(graphics)
```

```
## Examples from Venables & Ripley
```

```
spectrum(ldeaths)
spectrum(ldeaths, spans = c(3,5))
spectrum(ldeaths, spans = c(5,7))
spectrum(mdeaths, spans = c(3,3))
spectrum(fdeaths, spans = c(3,3))
```

```
## bivariate example
```

```
mfdeaths.spc <- spec.pgram(ts.union(mdeaths, fdeaths), spans = c(3,3))
# plots marginal spectra: now plot coherency and phase
plot(mfdeaths.spc, plot.type = "coherency")
plot(mfdeaths.spc, plot.type = "phase")
```

```
## now impose a lack of alignment
```

```
mfdeaths.spc <- spec.pgram(ts.intersect(mdeaths, lag(fdeaths, 4)),
  spans = c(3,3), plot = FALSE)
plot(mfdeaths.spc, plot.type = "coherency")
plot(mfdeaths.spc, plot.type = "phase")
```

```
stocks.spc <- spectrum(EuStockMarkets, kernel("daniell", c(30,50)),
  plot = FALSE)
plot(stocks.spc, plot.type = "marginal") # the default type
plot(stocks.spc, plot.type = "coherency")
plot(stocks.spc, plot.type = "phase")
```

```
sales.spc <- spectrum(ts.union(BJsales, BJsales.lead),  
                      kernel("modified.daniell", c(5,7)))  
plot(sales.spc, plot.type = "coherency")  
plot(sales.spc, plot.type = "phase")
```

`spec.taper`

```
spec.taper(x, p = 0.1)
```

`x`

`p`

```
p[i]x[, i]
```

[spec.pgram](#)[cpgm](#)

`spectrum`

`spectrum`

```
spectrum(x, ..., method = c("pgram", "ar"))
```

`x`

`method` `"pgram" "ar"`

`...` `plot.spec`

`spectrum`[spec.pgram](#)[spec.ar](#)

`1/frequency(x)(-frequency(x)/2, +frequency(x)/2]2π(-0.5,0.5](-π,π]`

`plot.spec`

"spec"

freq

spec freq

coh $\text{NULL } i + (j - 1) * (j - 2) / 2 \text{coh} i j x i < j$

phase NULLcoh

series

snames

method

plot

"spec"

[spec.arspec.pgramplot.spec](#)

```
require(graphics)
```

```
## Examples from Venables & Ripley
```

```
## spec.pgram
```

```
par(mfrow = c(2,2))
```

```
spectrum(lh)
```

```
spectrum(lh, spans = 3)
```

```
spectrum(lh, spans = c(3,3))
```

```
spectrum(lh, spans = c(3,5))
```

```
spectrum(ldeaths)
```

```
spectrum(ldeaths, spans = c(3,3))
```

```
spectrum(ldeaths, spans = c(3,5))
```

```
spectrum(ldeaths, spans = c(5,7))
```

```
spectrum(ldeaths, spans = c(5,7), log = "dB", ci = 0.8)
```

```
# for multivariate examples see the help for spec.pgram
```

```
## spec.ar
```

```
spectrum(lh, method = "ar")
```

```
spectrum(ldeaths, method = "ar")
```

splinefun

```
splinefun(x, y = NULL,
          method = c("fmm", "periodic", "natural", "monoH.FC", "hyman"),
          ties = mean)
```

```
spline(x, y = NULL, n = 3*length(x), method = "fmm",
       xmin = min(x), xmax = max(x), xout, ties = mean)
```

```
splinefunH(x, y, m)
```

xy	xy.coords ymethod = "hyman"
m	splinefunH() $m_i(x_i, y_i)$
method	"fmm""natural""periodic""monoH.FC""hyman"
n	xoutnxminxmax
xminxmax	xout
xout	
ties	x"ordered" listapprox

```
(x, y)method = "fmm"method = "natural"method = "periodic"
"monoH.FC"( $x_i, y_i, m_i$ )
"hyman"method = "fmm"
xmethod = "fmm"
```

```
splindexy
```

```
splinefunxderivderivderivx
```

```
splinefun
```

[approxapproxfun](#)
[interpSplineperiodicSpline](#)
[smooth.spline](#)

```
require(graphics)

op <- par(mfrow = c(2,1), mgp = c(2,.8,0), mar = 0.1+c(3,3,3,1))
n <- 9
x <- 1:n
y <- rnorm(n)
plot(x, y, main = paste("spline[fun](.) through", n, "points"))
lines(spline(x, y))
lines(spline(x, y, n = 201), col = 2)

y <- (x-6)^2
plot(x, y, main = "spline(.) -- 3 methods")
lines(spline(x, y, n = 201), col = 2)
lines(spline(x, y, n = 201, method = "natural"), col = 3)
lines(spline(x, y, n = 201, method = "periodic"), col = 4)
legend(6, 25, c("fmm", "natural", "periodic"), col = 2:4, lty = 1)

y <- sin((x-0.5)*pi)
f <- splinefun(x, y)
ls(envir = environment(f))
splinecoef <- get("z", envir = environment(f))
curve(f(x), 1, 10, col = "green", lwd = 1.5)
points(splinecoef, col = "purple", cex = 2)
curve(f(x, deriv = 1), 1, 10, col = 2, lwd = 1.5)
curve(f(x, deriv = 2), 1, 10, col = 2, lwd = 1.5, n = 401)
curve(f(x, deriv = 3), 1, 10, col = 2, lwd = 1.5, n = 401)
par(op)

## Manual spline evaluation --- demo the coefficients :
.x <- splinecoef$x
u <- seq(3, 6, by = 0.25)
(ii <- findInterval(u, .x))
dx <- u - .x[ii]
f.u <- with(splinecoef,
            y[ii] + dx*(b[ii] + dx*(c[ii] + dx* d[ii])))
stopifnot(all.equal(f(u), f.u))

## An example with ties (non-unique x values):
set.seed(1); x <- round(rnorm(30), 1); y <- sin(pi * x) + rnorm(30)/10
plot(x, y, main = "spline(x,y) when x has ties")
```

```

lines(spline(x, y, n = 201), col = 2)
## visualizes the non-unique ones:
tx <- table(x); mx <- as.numeric(names(tx[tx > 1]))
ry <- matrix(unlist(tapply(y, match(x, mx), range, simplify = FALSE)),
             ncol = 2, byrow = TRUE)
segments(mx, ry[, 1], mx, ry[, 2], col = "blue", lwd = 2)

## Another example with sorted x, but ties:
set.seed(8); x <- sort(round(rnorm(30), 1)); y <- round(sin(pi * x) + rnorm(30)/10, 3)
summary(diff(x) == 0) # -> 7 duplicated x-values
str(spline(x, y, n = 201, ties="ordered")) # all '$y' entries are NaN
## The default (ties=mean) is ok, but most efficient to use instead is
sxyo <- spline(x, y, n = 201, ties= list("ordered", mean))
sapply(sxyo, summary)# all fine now
plot(x, y, main = "spline(x,y, ties=list(\"ordered\", mean)) for when x has ties")
lines(sxyo, col="blue")

## An example of monotone interpolation
n <- 20
set.seed(11)
x. <- sort(runif(n)) ; y. <- cumsum(abs(rnorm(n)))
plot(x., y.)
curve(splinefun(x., y.)(x), add = TRUE, col = 2, n = 1001)
curve(splinefun(x., y., method = "monoH.FC")(x), add = TRUE, col = 3, n = 1001)
curve(splinefun(x., y., method = "hyman")(x), add = TRUE, col = 4, n = 1001)
legend("topleft",
      paste0("splinefun( \"", c("fmm", "monoH.FC", "hyman"), "\" )"),
      col = 2:4, lty = 1, bty = "n")

## and one from Fritsch and Carlson (1980), Dougherty et al (1989)
x. <- c(7.09, 8.09, 8.19, 8.7, 9.2, 10, 12, 15, 20)
f <- c(0, 2.76429e-5, 4.37498e-2, 0.169183, 0.469428, 0.943740,
      0.998636, 0.999919, 0.999994)
s0 <- splinefun(x., f)
s1 <- splinefun(x., f, method = "monoH.FC")
s2 <- splinefun(x., f, method = "hyman")
plot(x., f, ylim = c(-0.2, 1.2))
curve(s0(x), add = TRUE, col = 2, n = 1001) -> m0
curve(s1(x), add = TRUE, col = 3, n = 1001)
curve(s2(x), add = TRUE, col = 4, n = 1001)
legend("right",
      paste0("splinefun( \"", c("fmm", "monoH.FC", "hyman"), "\" )"),
      col = 2:4, lty = 1, bty = "n")

## they seem identical, but are not quite:
xx <- m0$x
plot(xx, s1(xx) - s2(xx), type = "l", col = 2, lwd = 2,
     main = "Difference monoH.FC - hyman"); abline(h = 0, lty = 3)

x <- xx[xx < 10.2] ## full range: x <- xx .. does not show enough
ccol <- adjustcolor(2:4, 0.8)
matplot(x, cbind(s0(x, deriv = 2), s1(x, deriv = 2), s2(x, deriv = 2))^2,
        lwd = 2, col = ccol, type = "l", ylab = quote({{f*second}(x)}^2),
        main = expression({{f*second}(x)}^2 ~" for the three 'splines'"))
legend("topright",
      paste0("splinefun( \"", c("fmm", "monoH.FC", "hyman"), "\" )"),
      lwd = 2, col = ccol, lty = 1:3, bty = "n")

```

```
## --> "hyman" has slightly smaller Integral f''(x)^2 dx than "FC",
## here, and both are 'much worse' than the regular fmm spline.
```

SSasymp	nls
---------	-----

```
selfStartinitialAsymR0lrc
SSweibull()
```

```
SSasymp(input, Asym, R0, lrc)
```

```
input
Asym      input
R0        input
lrc
```

```
inputAsym+(R0-Asym)*exp(-exp(lrc)*input)AsymR0lrcgradient
```

```
nlselfStart
```

```
Lob.329 <- Loblolly[ Loblolly$Seed == "329", ]
SSasymp( Lob.329$age, 100, -8.5, -3.2 ) # response only
local({
  Asym <- 100 ; resp0 <- -8.5 ; lrc <- -3.2
  SSasymp( Lob.329$age, Asym, resp0, lrc) # response _and_ gradient
})
getInitial(height ~ SSasymp( age, Asym, resp0, lrc), data = Lob.329)
## Initial values are in fact the converged values
fm1 <- nls(height ~ SSasymp( age, Asym, resp0, lrc), data = Lob.329)
summary(fm1)

## Visualize the SSasymp() model parametrization :

xx <- seq(-.3, 5, length.out = 101)
## Asym + (R0-Asym) * exp(-exp(lrc)* x) :
yy <- 5 - 4 * exp(-xx / exp(3/4))
stopifnot( all.equal(yy, SSasymp(xx, Asym = 5, R0 = 1, lrc = -3/4)) )
require(graphics)
op <- par(mar = c(0, .2, 4.1, 0))
plot(xx, yy, type = "l", axes = FALSE, ylim = c(0,5.2), xlim = c(-.3, 5),
      xlab = "", ylab = "", lwd = 2,
      main = quote("Parameters in the SSasymp model " ~
```

```

      {f[phi](x) == phi[1] + (phi[2]-phi[1])*~e^{~e^{phi[3]}*~x}}))
mtext(quote(list(phi[1] == "Asym", phi[2] == "R0", phi[3] == "lrc")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = 5, lty = 3)
arrows(c(0.35, 0.65), 1,
      c(0, 1), 1, length = 0.08, angle = 25); text(0.5, 1, quote(1))
y0 <- 1 + 4*exp(-3/4) ; t.5 <- log(2) / exp(-3/4) ; AR2 <- 3 # (Asym + R0)/2
segments(c(1, 1), c(1, y0),
      c(1, 0), c(y0, 1), lty = 2, lwd = 0.75)
text(1.1, 1/2+y0/2, quote((phi[1]-phi[2])*e^phi[3]), adj = c(0,.5))
axis(2, at = c(1,AR2,5), labels= expression(phi[2], frac(phi[1]+phi[2],2), phi[1]),
      pos=0, las=1)
arrows(c(.6,t.5-.6), AR2,
      c(0, t.5), AR2, length = 0.08, angle = 25)
text(t.5/2, AR2, quote(t[0.5]))
text(t.5+.4, AR2,
      quote({f(t[0.5]) == frac(phi[1]+phi[2],2)}~{} %=>% {}~~
      {t[0.5] == frac(log(2), e^{phi[3]})}), adj = c(0, 0.5))
par(op)

```

SSasympOff

nls

selfStartinitialAsymlrcc0

SSasympOff(input, Asym, lrc, c0)

input

Asym input

lrc

c0 input

inputAsym*(1 - exp(-exp(lrc)*(input - c0)))Asymlrcc0gradient

[nlsselfStart](#)example(SSasympOff)SSasympOff

```

C02.Qn1 <- C02[C02$Plant == "Qn1", ]
SSasymptOff(C02.Qn1$conc, 32, -4, 43) # response only
local({ Asym <- 32; lrc <- -4; c0 <- 43
  SSasymptOff(C02.Qn1$conc, Asym, lrc, c0) # response and gradient
})
getInitial(uptake ~ SSasymptOff(conc, Asym, lrc, c0), data = C02.Qn1)
## Initial values are in fact the converged values
fm1 <- nls(uptake ~ SSasymptOff(conc, Asym, lrc, c0), data = C02.Qn1)
summary(fm1)

## Visualize the SSasymptOff() model parametrization :

xx <- seq(0.25, 8, by=1/16)
yy <- 5 * (1 - exp(-(xx - 3/4)*0.4))
stopifnot( all.equal(yy, SSasymptOff(xx, Asym = 5, lrc = log(0.4), c0 = 3/4)) )
require(graphics)
op <- par(mar = c(0, 0, 4.0, 0))
plot(xx, yy, type = "l", axes = FALSE, ylim = c(-.5,6), xlim = c(-1, 8),
      xlab = "", ylab = "", lwd = 2,
      main = "Parameters in the SSasymptOff model")
mtext(quote(list(phi[1] == "Asym", phi[2] == "lrc", phi[3] == "c0")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = 5, lty = 3)
arrows(-0.8, c(2.1, 2.9),
       -0.8, c(0, 5), length = 0.1, angle = 25)
text(-0.8, 2.5, quote(phi[1]))
segments(3/4, -.2, 3/4, 1.6, lty = 2)
text(3/4, c(-.3, 1.7), quote(phi[3]))
arrows(c(1.1, 1.4), -.15,
       c(3/4, 7/4), -.15, length = 0.07, angle = 25)
text(3/4 + 1/2, -.15, quote(1))
segments(c(3/4, 7/4, 7/4), c(0, 0, 2), # 5 * exp(log(0.4)) = 2
       c(7/4, 7/4, 3/4), c(0, 2, 0), lty = 2, lwd = 2)
text(7/4 + .1, 2./2, quote(phi[1]*e^phi[2]), adj = c(0, .5))
par(op)

```

SSasymptOrig

nls

[selfStart](#)initialAsymlrc

SSasymptOrig(input, Asym, lrc)

input

Asym

lrc

```
inputAsym*(1 - exp(-exp(lrc)*input))Asymlrcgradient
```

[nlselfStart](#)

```
Lob.329 <- Loblolly[ Loblolly$Seed == "329", ]
SSasypOrig(Lob.329$age, 100, -3.2) # response only
local({ Asym <- 100; lrc <- -3.2
  SSasypOrig(Lob.329$age, Asym, lrc) # response and gradient
})
getInitial(height ~ SSasypOrig(age, Asym, lrc), data = Lob.329)
## Initial values are in fact the converged values
fm1 <- nls(height ~ SSasypOrig(age, Asym, lrc), data = Lob.329)
summary(fm1)

## Visualize the SSasypOrig() model parametrization :

xx <- seq(0, 5, length.out = 101)
yy <- 5 * (1- exp(-xx * log(2)))
stopifnot( all.equal(yy, SSasypOrig(xx, Asym = 5, lrc = log(log(2)))) )

require(graphics)
op <- par(mar = c(0, 0, 3.5, 0))
plot(xx, yy, type = "l", axes = FALSE, ylim = c(0,5), xlim = c(-1/4, 5),
      xlab = "", ylab = "", lwd = 2,
      main = quote("Parameters in the SSasypOrig model"~~ f[phi](x)))
mtext(quote(list(phi[1] == "Asym", phi[2] == "lrc")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = 5, lty = 3)
axis(2, at = 5*c(1/2,1), labels= expression(frac(phi[1],2), phi[1]), pos=0, las=1)
arrows(c(.3,.7), 5/2,
       c(0, 1 ), 5/2, length = 0.08, angle = 25)
text( 0.5, 5/2, quote(t[0.5]))
text( 1+.4, 5/2,
      quote({f(t[0.5]) == frac(phi[1],2)}~{ }%=>% { }~~{t[0.5] == frac(log(2), e^{phi[2]}}}),
      adj = c(0, 0.5))
par(op)
```

SSbiexp

nls

selfStartinitialA1lrc1A2lrc2

```
SSbiexp(input, A1, lrc1, A2, lrc2)
```

```
input
```

```
A1
```

```
lrc1
```

```
A2
```

```
lrc2
```

```
inputA1*exp(-exp(lrc1)*input)+A2*exp(-exp(lrc2)*input)A1lrc1A2lrc2gradient
```

```
nlselfStart
```

```
Indo.1 <- Indometh[Indometh$Subject == 1, ]
SSbiexp( Indo.1$time, 3, 1, 0.6, -1.3 ) # response only
A1 <- 3; lrc1 <- 1; A2 <- 0.6; lrc2 <- -1.3
SSbiexp( Indo.1$time, A1, lrc1, A2, lrc2 ) # response and gradient
print(getInitial(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = Indo.1),
      digits = 5)
## Initial values are in fact the converged values
fm1 <- nls(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = Indo.1)
summary(fm1)

## Show the model components visually
require(graphics)

xx <- seq(0, 5, length.out = 101)
y1 <- 3.5 * exp(-4*xx)
y2 <- 1.5 * exp(-xx)
plot(xx, y1 + y2, type = "l", lwd=2, ylim = c(-0.2,6), xlim = c(0, 5),
     main = "Components of the SSbiexp model")
lines(xx, y1, lty = 2, col="tomato"); abline(v=0, h=0, col="gray40")
lines(xx, y2, lty = 3, col="blue2" )
legend("topright", c("y1+y2", "y1 = 3.5 * exp(-4*x)", "y2 = 1.5 * exp(-x)"),
      lty=1:3, col=c("black","tomato","blue2"), bty="n")
axis(2, pos=0, at = c(3.5, 1.5), labels = c("A1","A2"), las=2)

## and how you could have got their sum via SSbiexp():
ySS <- SSbiexp(xx, 3.5, log(4), 1.5, log(1))
##          ---          ---
stopifnot(all.equal(y1+y2, ySS, tolerance = 1e-15))

## Show a no-noise example
datN <- data.frame(time = (0:600)/64)
datN$conc <- predict(fm1, newdata=datN)
```



```

plot(conc ~ time, data=datN) # perfect, no noise

## Fails by default (scaleOffset=0) on most platforms {also after increasing maxiter !}
## Not run:
      nls(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = datN, trace=TRUE)
## End(Not run)

fmX1 <- nls(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = datN,
            control = list(scaleOffset=1))
fmX  <- nls(conc ~ SSbiexp(time, A1, lrc1, A2, lrc2), data = datN,
            control = list(scaleOffset=1, printEval=TRUE, tol=1e-11, nDcentral=TRUE), trace=TRUE)
all.equal(coef(fm1), coef(fmX1), tolerance=0) # ... rel.diff.: 1.57e-6
all.equal(coef(fm1), coef(fmX),  tolerance=0) # ... rel.diff.: 1.03e-12

stopifnot(all.equal(coef(fm1), coef(fmX1), tolerance = 6e-6),
          all.equal(coef(fm1), coef(fmX ), tolerance = 1e-11))

```

SSD

```

# S3 method for class 'mlm'
SSD(object, ...)

# S3 methods for class 'SSD' and 'mlm'
estVar(object, ...)

```

```

object      object"mlm""SSD"estVar
...

```

```
SSD()"SSD"
```

```
SSD
```

```
df
```

```
call      object
```

```
estVar
```

[mauchly.testanova.mlm](#)

```

# Lifted from Baron+Li:
# "Notes on the use of R for psychology experiments and questionnaires"
# Maxwell and Delaney, p. 497
reacttime <- matrix(c(
  420, 420, 480, 480, 600, 780,
  420, 480, 480, 360, 480, 600,
  480, 480, 540, 660, 780, 780,
  420, 540, 540, 480, 780, 900,
  540, 660, 540, 480, 660, 720,
  360, 420, 360, 360, 480, 540,
  480, 480, 600, 540, 720, 840,
  480, 600, 660, 540, 720, 900,
  540, 600, 540, 480, 720, 780,
  480, 420, 540, 540, 660, 780),
  ncol = 6, byrow = TRUE,
  dimnames = list(subj = 1:10,
    cond = c("deg0NA", "deg4NA", "deg8NA",
      "deg0NP", "deg4NP", "deg8NP")))

mlmfit <- lm(reacttime ~ 1)
SSD(mlmfit)
estVar(mlmfit)

```

SSfol	nls
-------	-----

```
selfStartinitiallKelKalCl
```

```
SSfol(Dose, input, lKe, lKa, lCl)
```

```
Dose
```

```
input
```

```
lKe
```

```
lKa
```

```
lCl
```

```
input
```

```
Dose * exp(lKe+lKa-lCl) * (exp(-exp(lKe)*input) - exp(-exp(lKa)*input))
/ (exp(lKa) - exp(lKe))
```

```
lKelKalClgradient
```

nlsselfStart

```
Theoph.1 <- Theoph[ Theoph$Subject == 1, ]
with(Theoph.1, SSfol(Dose, Time, -2.5, 0.5, -3)) # response only
with(Theoph.1, local({ lKe <- -2.5; lKa <- 0.5; lCl <- -3
  SSfol(Dose, Time, lKe, lKa, lCl) # response _and_ gradient
}))
getInitial(conc ~ SSfol(Dose, Time, lKe, lKa, lCl), data = Theoph.1)
## Initial values are in fact the converged values
fm1 <- nls(conc ~ SSfol(Dose, Time, lKe, lKa, lCl), data = Theoph.1)
summary(fm1)
```

SSfpl

nls

selfStartinitialABxmidscal

SSfpl(input, A, B, xmid, scal)

input

A input

B input

xmid inputSSfplABxmid

scal input

inputA+(B-A)/(1+exp((xmid-input)/scal))ABxmidscalgradient

nlsselfStart

```
Chick.1 <- ChickWeight[ChickWeight$Chick == 1, ]
SSfpl(Chick.1$Time, 13, 368, 14, 6) # response only
local({
  A <- 13; B <- 368; xmid <- 14; scal <- 6
  SSfpl(Chick.1$Time, A, B, xmid, scal) # response _and_ gradient
})
print(getInitial(weight ~ SSfpl(Time, A, B, xmid, scal), data = Chick.1),
  digits = 5)
## Initial values are in fact the converged values
fm1 <- nls(weight ~ SSfpl(Time, A, B, xmid, scal), data = Chick.1)
```

```
summary(fm1)

## Visualizing the SSfpl() parametrization
xx <- seq(-0.5, 5, length.out = 101)
yy <- 1 + 4 / (1 + exp((2-xx))) # == SSfpl(xx, *) :
stopifnot( all.equal(yy, SSfpl(xx, A = 1, B = 5, xmid = 2, scal = 1)) )
require(graphics)
op <- par(mar = c(0, 0, 3.5, 0))
plot(xx, yy, type = "l", axes = FALSE, ylim = c(0,6), xlim = c(-1, 5),
      xlab = "", ylab = "", lwd = 2,
      main = "Parameters in the SSfpl model")
mtext(quote(list(phi[1] == "A", phi[2] == "B", phi[3] == "xmid", phi[4] == "scal")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = c(1, 5), lty = 3)
arrows(-0.8, c(2.1, 2.9),
       -0.8, c(0, 5 ), length = 0.1, angle = 25)
text (-0.8, 2.5, quote(phi[1]))
arrows(-0.3, c(1/4, 3/4),
       -0.3, c(0, 1 ), length = 0.07, angle = 25)
text (-0.3, 0.5, quote(phi[2]))
text(2, -.1, quote(phi[3]))
segments(c(2,3,3), c(0,3,4), # SSfpl(x = xmid = 2) = 3
         c(2,3,2), c(3,4,3), lty = 2, lwd = 0.75)
arrows(c(2.3, 2.7), 3,
       c(2.0, 3 ), 3, length = 0.08, angle = 25)
text( 2.5, 3, quote(phi[4])); text(3.1, 3.5, "1")
par(op)
```

SSgompertz

nls

selfStartinitialAsymb2b3

SSgompertz(x, Asym, b2, b3)

x

Asym

b2 x = 0

b3 x

inputAsym*exp(-b2*b3^x)Asymb2b3gradient

nlselfStart

```
DNase.1 <- subset(DNase, Run == 1)
SSgompertz(log(DNase.1$conc), 4.5, 2.3, 0.7) # response only
local({ Asym <- 4.5; b2 <- 2.3; b3 <- 0.7
  SSgompertz(log(DNase.1$conc), Asym, b2, b3) # response _and_ gradient
})
print(getInitial(density ~ SSgompertz(log(conc), Asym, b2, b3),
  data = DNase.1), digits = 5)
## Initial values are in fact the converged values
fm1 <- nls(density ~ SSgompertz(log(conc), Asym, b2, b3),
  data = DNase.1)
summary(fm1)
plot(density ~ log(conc), DNase.1, # xlim = c(0, 21),
  main = "SSgompertz() fit to DNase.1")
ux <- par("usr")[1:2]; x <- seq(ux[1], ux[2], length.out=250)
lines(x, do.call(SSgompertz, c(list(x=x), coef(fm1))), col = "red", lwd=2)
As <- coef(fm1)[["Asym"]]; abline(v = 0, h = 0, lty = 3)
axis(2, at= exp(-coef(fm1)[["b2"]]), quote(e^{-b[2]}), las=1, pos=0)
```

SSlogis

nls

selfStartinitialAsymxmidscalmin(input)

SSlogis(input, Asym, xmid, scal)

input

Asym

xmid xSSlogisAsym/2xmid

scal input

inputAsym/(1+exp((xmid-input)/scal))Asymxmidscalgradient

nlselfStart

```

Chick.1 <- ChickWeight[ChickWeight$Chick == 1, ]
SSlogis(Chick.1$Time, 368, 14, 6) # response only
local({
  Asym <- 368; xmid <- 14; scal <- 6
  SSlogis(Chick.1$Time, Asym, xmid, scal) # response _and_ gradient
})
getInitial(weight ~ SSlogis(Time, Asym, xmid, scal), data = Chick.1)
## Initial values are in fact the converged one here, "Number of iter...: 0" :
fm1 <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = Chick.1)
summary(fm1)
## but are slightly improved here:
fm2 <- update(fm1, control=nls.control(tol = 1e-9, warnOnly=TRUE), trace = TRUE)
all.equal(coef(fm1), coef(fm2)) # "Mean relative difference: 9.6e-6"
str(fm2$convInfo) # 3 iterations

dwlgl1 <- data.frame(Prop = c(rep(0,5), 2, 5, rep(9, 9)), end = 1:16)
iPar <- getInitial(Prop ~ SSlogis(end, Asym, xmid, scal), data = dwlgl1)
## failed in R <= 3.4.2 (because of the '0's in 'Prop')
stopifnot(all.equal(tolerance = 1e-6,
  iPar, c(Asym = 9.0678, xmid = 6.79331, scal = 0.499934)))

## Visualize the SSlogis() model parametrization :
xx <- seq(-0.75, 5, by=1/32)
yy <- 5 / (1 + exp((2-xx)/0.6)) # == SSlogis(xx, *):
stopifnot( all.equal(yy, SSlogis(xx, Asym = 5, xmid = 2, scal = 0.6)) )
require(graphics)
op <- par(mar = c(0.5, 0, 3.5, 0))
plot(xx, yy, type = "l", axes = FALSE, ylim = c(0,6), xlim = c(-1, 5),
  xlab = "", ylab = "", lwd = 2,
  main = "Parameters in the SSlogis model")
mtext(quote(list(phi[1] == "Asym", phi[2] == "xmid", phi[3] == "scal")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = 5, lty = 3)
arrows(-0.8, c(2.1, 2.9),
  -0.8, c(0, 5 ), length = 0.1, angle = 25)
text (-0.8, 2.5, quote(phi[1]))
segments(c(2,2.6,2.6), c(0, 2.5,3.5), # NB. SSlogis(x = xmid = 2) = 2.5
  c(2,2.6,2 ), c(2.5,3.5,2.5), lty = 2, lwd = 0.75)
text(2, -.1, quote(phi[2]))
arrows(c(2.2, 2.4), 2.5,
  c(2.0, 2.6), 2.5, length = 0.08, angle = 25)
text( 2.3, 2.5, quote(phi[3])); text(2.7, 3, "1")
par(op)

```

SSmicmen

nls

selfStartinitialVmK

```
SSmicmen(input, Vm, K)
```

```
input
```

```
Vm
```

```
K          input
```

```
inputVm*input/(K+input)VmKgradient
```

```
nlselfStart
```

```
PurTrt <- Puromycin[ Puromycin$state == "treated", ]
SSmicmen(PurTrt$conc, 200, 0.05) # response only
local({ Vm <- 200; K <- 0.05
  SSmicmen(PurTrt$conc, Vm, K) # response _and_ gradient
})
print(getInitial(rate ~ SSmicmen(conc, Vm, K), data = PurTrt), digits = 3)
## Initial values are in fact the converged values
fm1 <- nls(rate ~ SSmicmen(conc, Vm, K), data = PurTrt)
summary(fm1)
## Alternative call using the subset argument
fm2 <- nls(rate ~ SSmicmen(conc, Vm, K), data = Puromycin,
  subset = state == "treated")
summary(fm2) # The same indeed:
stopifnot(all.equal(coef(summary(fm1)), coef(summary(fm2))))

## Visualize the SSmicmen() Michaelis-Menton model parametrization :

xx <- seq(0, 5, length.out = 101)
yy <- 5 * xx/(1+xx)
stopifnot(all.equal(yy, SSmicmen(xx, Vm = 5, K = 1)))
require(graphics)
op <- par(mar = c(0, 0, 3.5, 0))
plot(xx, yy, type = "l", lwd = 2, ylim = c(-1/4,6), xlim = c(-1, 5),
  ann = FALSE, axes = FALSE, main = "Parameters in the SSmicmen model")
mtext(quote(list(phi[1] == "Vm", phi[2] == "K")))
usr <- par("usr")
arrows(usr[1], 0, usr[2], 0, length = 0.1, angle = 25)
arrows(0, usr[3], 0, usr[4], length = 0.1, angle = 25)
text(usr[2] - 0.2, 0.1, "x", adj = c(1, 0))
text(-0.1, usr[4], "y", adj = c(1, 1))
abline(h = 5, lty = 3)
arrows(-0.8, c(2.1, 2.9),
  -0.8, c(0, 5), length = 0.1, angle = 25)
text(-0.8, 2.5, quote(phi[1]))
segments(1, 0, 1, 2.7, lty = 2, lwd = 0.75)
text(1, 2.7, quote(phi[2]))
par(op)
```

SSweibull

nls

selfStartinitialAsymDropIrcpwr

SSweibull(x, Asym, Drop, lrc, pwr)

x

Asym x

Drop Asymy

lrc

pwr x

[SSasymp](#)SSasymppwr

xAsym-Drop*exp(-exp(lrc)*x^pwr)AsymDropIrcpwrgradient

[nlselfStartSSasymp](#)

```
Chick.6 <- subset(ChickWeight, (Chick == 6) & (Time > 0))
SSweibull(Chick.6$Time, 160, 115, -5.5, 2.5) # response only
local({ Asym <- 160; Drop <- 115; lrc <- -5.5; pwr <- 2.5
  SSweibull(Chick.6$Time, Asym, Drop, lrc, pwr) # response _and_ gradient
})

getInitial(weight ~ SSweibull(Time, Asym, Drop, lrc, pwr), data = Chick.6)
## Initial values are in fact the converged values
fm1 <- nls(weight ~ SSweibull(Time, Asym, Drop, lrc, pwr), data = Chick.6)
summary(fm1)

## Data and Fit:
plot(weight ~ Time, Chick.6, xlim = c(0, 21), main = "SSweibull() fit to Chick.6")
ux <- par("usr")[1:2]; x <- seq(ux[1], ux[2], length.out=250)
lines(x, do.call(SSweibull, c(list(x=x), coef(fm1))), col = "red", lwd=2)
As <- coef(fm1)[["Asym"]]; abline(v = 0, h = c(As, As - coef(fm1)[["Drop"]]), lty = 3)
```

start

```
start(x, ...)
end(x, ...)
```

```
x
...
```

```
tsp $x_{1995.5}c(1995, 7)f_{n+i}/f_c(n, i+1)i = 0f = 1$ 
```

startend

tstimetsp

stat.anova

```
lmglm $anova(\dots, test \neq NULL)$ 
```

```
stat.anova(table, test = c("Rao", "LRT", "Chisq", "F", "Cp"),
            scale, df.scale, n)
```

```
table       $anova.glm(\dots, test = NULL)$ 
test       "Rao" "LRT" "Chisq" "F" "Cp"
scale
df.scale   scale
n
```

tabletest

[anova.lmanova.glm](#)

```
##-- Continued from '?glm':

print(ag <- anova(glm.D93))
stat.anova(ag$table, test = "Cp",
           scale = sum(resid(glm.D93, "pearson")^2)/4,
           df.scale = 4, n = 9)
```

stats-deprecated

[Deprecated](#)

step

```
step(object, scope, scale = 0,
      direction = c("both", "backward", "forward"),
      trace = 1, keep = NULL, steps = 1000, k = 2, ...)
```

object	"lm" "glm"
scope	upperlower
scale	lmaovglm0extractAIC
direction	"both" "backward" "forward" "both" scopedirection "backward"
trace	step
keep	AICkeep
steps	
k	k = 2k = log(n)
...	extractAIC

```
stepadd1drop1extractAIC $C_p$ 
scopelowerupperscopeupperlowerupper
scopeobjectupdate.formula.scope.^2
glm scale"glm"extractAICgaussianbinomialpoissonsscalescale
```

```
"anova""keep"keep="Resid. Dev"lmaovsurvreg
```

```
na.action = na.omit
nobs
```

```
stepAIC
```

```
stepstepAIC
step
```

```
stepAICadd1drop1
```

```
## following on from example(lm)

step(lm.D9)

summary(lm1 <- lm(Fertility ~ ., data = swiss))
slm1 <- step(lm1)
summary(slm1)
slm1$anova
```

stepfun

$(x_1, \dots, x_n)(y_0, y_1, \dots, y_n)$
 $\text{stepfun}(x, y, \dots) \text{fn}$
 $f_n(t) = c_i t \in (x_i, x_{i+1}) \text{right} = \text{FALSE}$
 $f_n(x_i) = y_i \text{right} = \text{TRUE} f_n(x_i) = y_{i-1} i = 1, \dots, n$
 $c_i \text{fright} = \text{FALSE}, f = 0 \text{fn} c_i y c_i = (1 - f)y_i + f \cdot y_{i+1} \text{fn} \text{right} = \text{TRUE}, f = 1$

```
stepfun(x, y, f = as.numeric(right), ties = "ordered",  
        right = FALSE)
```

```
is.stepfun(x)  
knots(Fn, ...)  
as.stepfun(x, ...)
```

```
## S3 method for class 'stepfun'  
print(x, digits = getOption("digits") - 2, ...)
```

```
## S3 method for class 'stepfun'  
summary(object, ...)
```

```
x          stepfun()xobject  
y          x  
f          approxfun  
ties       x"ordered"approxfun  
right  
Fobject    "stepfun"  
digits     print  
...
```

```
"stepfun"fn  
"summary(.)""print(.)""plot(.)"plot.stepfun"stepfun"  
environmentfn  
"x""y"  
"n"  
"f"  
"yleft""yright"  
  
"method"   == "constant"approxfun(.)  
knots(fn)
```

"stepfun"

```
eval(attr(old_obj, "call"), environment(old_obj))
```

<maechler@stat.math.ethz.ch>

[ecdfplot.stepfun](#)

[approxfun splinefun](#)

```
y0 <- c(1., 2., 4., 3.)
sfun0 <- stepfun(1:3, y0, f = 0)
sfun.2 <- stepfun(1:3, y0, f = 0.2)
sfun1 <- stepfun(1:3, y0, f = 1)
sfun1c <- stepfun(1:3, y0, right = TRUE) # hence f=1
sfun0
summary(sfun0)
summary(sfun.2)

## look at the internal structure:
unclass(sfun0)
ls(envir = environment(sfun0))

x0 <- seq(0.5, 3.5, by = 0.25)
rbind(x = x0, f.f0 = sfun0(x0), f.f02 = sfun.2(x0),
      f.f1 = sfun1(x0), f.f1c = sfun1c(x0))
## Identities :
stopifnot(identical(y0[-1], sfun0(1:3)), # right = FALSE
          identical(y0[-4], sfun1c(1:3))) # right = TRUE
```

stl

loess

```
stl(x, s.window, s.degree = 0,
    t.window = NULL, t.degree = 1,
    l.window = nextodd(period), l.degree = t.degree,
    s.jump = ceiling(s.window/10),
    t.jump = ceiling(t.window/10),
    l.jump = ceiling(l.window/10),
    robust = FALSE,
    inner = if(robust) 1 else 2,
    outer = if(robust) 15 else 0,
    na.action = na.fail)
```

```
x            "ts"
s.window     "periodic"
s.degree
t.window     NULLnextodd(ceiling((1.5*period) / (1-(1.5/s.window))))
t.degree
l.window     frequency(x)
l.degree
s.jumplt.jump*.jump
robust       loess
inner
outer
na.action
```

```
s.window = "periodic"remainder
"stl"plot.stl
```

```
stl"stl"
time.series  seasonaltrendremainder
weights
call
win          "s""t""l"
deg
jump
ni
no
```

```
netlib
```

```
plot.stlstoessstl
StructTS
```

```

require(graphics)

plot(stl(nottem, "per"))
plot(stl(nottem, s.window = 7, t.window = 50, t.jump = 1))

plot(stllc <- stl(log(co2), s.window = 21))
summary(stllc)
## linear trend, strict period.
plot(stl(log(co2), s.window = "per", t.window = 1000))

## Two STL plotted side by side :
  stmd <- stl(mdeaths, s.window = "per") # non-robust
summary(stmR <- stl(mdeaths, s.window = "per", robust = TRUE))
op <- par(mar = c(0, 4, 0, 3), oma = c(5, 0, 4, 0), mfcol = c(4, 2))
plot(stmd, set.pars = NULL, labels = NULL,
      main = "stl(mdeaths, s.w = \"per\", robust = FALSE / TRUE )")
plot(stmR, set.pars = NULL)
# mark the 'outliers' :
(i0 <- which(stmR $ weights < 1e-8)) # 10 were considered outliers
sts <- stmR$time.series
points(time(sts)[i0], 0.8* sts["remainder"][i0], pch = 4, col = "red")
par(op) # reset

```

stlmethods

stl**stl**plot

printsummary

```

## S3 method for class 'stl'
plot(x, labels = colnames(X),
      set.pars = list(mar = c(0, 6, 0, 6), oma = c(6, 0, 4, 0),
                      tck = -0.01, mfrow = c(nplot, 1)),
      main = NULL, range.bars = TRUE, ...,
      col.range = "light gray")

## S3 method for class 'stl'
summary(object, digits = max(3L, getOption("digits") - 3L), ...)

```

xobject	stl
labels	
set.pars	par (.)
main	
range.bars	
col.range	...
...	summary() z digits
digits	summary time.series w eight s

[plot.tsstlsummary.default](#)

StructTS

```
StructTS(x, type = c("level", "trend", "BSM"), init = NULL,  
         fixed = NULL, optim.control = NULL)
```

```
x  
type          frequency(x) > 1  
init  
fixed         NAfixed  
optim.control optim"L-BFGS-B"
```

type = "level" μ_t

$$\mu_{t+1} = \mu_t + \xi_t, \xi_t \sim N(0, \sigma_\xi^2)$$

$$x_t = \mu_t + \epsilon_t, \epsilon_t \sim N(0, \sigma_\epsilon^2)$$

$\sigma_\xi^2 \sigma_\epsilon^2$

type = "trend" μ_t

$$\mu_{t+1} = \mu_t + \nu_t + \xi_t, \xi_t \sim N(0, \sigma_\xi^2)$$

$$\nu_{t+1} = \nu_t + \zeta_t, \zeta_t \sim N(0, \sigma_\zeta^2)$$

$\sigma_\zeta^2 = 0$ $\sigma_\xi^2 = 0$

type = "BSM"

$$x_t = \mu_t + \gamma_t + \epsilon_t, \epsilon_t \sim N(0, \sigma_\epsilon^2)$$

γ_t

$$\gamma_{t+1} = -\gamma_t + \dots + \gamma_{t-s+2} + \omega_t, \omega_t \sim N(0, \sigma_\omega^2)$$

$\sigma_\omega^2 = 0$

"StructTS"

```
coef
loglik      arima
loglik0
data        x
residuals
fitted      t
call
series      x
code        convergenceoptim
modelmodel0 KalmanLikemodel0model
xtsp        tspx
```

[AirPassengersStructTS](#) σ_ϵ^2

[KalmanLiketsSmoothstl](#)

```
## see also JohnsonJohnson, Nile and AirPassengers
require(graphics)
```

```
trees <- window(treering, start = 0)
(fit <- StructTS(trees, type = "level"))
plot(trees)
lines(fitted(fit), col = "green")
tsdiag(fit)
```

```
(fit <- StructTS(log10(UKgas), type = "BSM"))
par(mfrow = c(4, 1)) # to give appropriate aspect ratio for next plot.
plot(log10(UKgas))
plot(cbind(fitted(fit), resid=resid(fit)), main = "UK gas consumption")
```

```
## keep some parameters fixed; trace optimizer:
StructTS(log10(UKgas), type = "BSM", fixed = c(0.1,0.001,NA,NA),
         optim.control = list(trace = TRUE))
```

summary.aov

```
## S3 method for class 'aov'
summary(object, intercept = FALSE, split,
        expand.split = TRUE, keep.zero.df = TRUE, ...)
```

```
## S3 method for class 'aovlist'
summary(object, ...)
```

```
object          "aov""aovlist"
intercept
split
expand.split
keep.zero.df
...             summary.aovlistsummary.aov
```

```
c("summary.aov", "listof")"summary.aovlist"
"anova""data.frame""Df""Sum Sq""Mean Sq""F value""Pr(>F)""Residuals"
```

```
expand.split = TRUEFALSE
```

[aovsummarymodel.tablesTukeyHSD](#)

```
## For a simple example see example(aov)

# Cochran and Cox (1957, p.164)
# 3x3 factorial with ordered factors, each is average of 12.
CC <- data.frame(
  y = c(449, 413, 326, 409, 358, 291, 341, 278, 312)/12,
  P = ordered(gl(3, 3)), N = ordered(gl(3, 1, 9))
)
CC.aov <- aov(y ~ N * P, data = CC , weights = rep(12, 9))
summary(CC.aov)

# Split both main effects into linear and quadratic parts.
summary(CC.aov, split = list(N = list(L = 1, Q = 2),
                              P = list(L = 1, Q = 2)))
```

```
# Split only the interaction
summary(CC.aov, split = list("N:P" = list(L.L = 1, Q = 2:4)))

# split on just one var
summary(CC.aov, split = list(P = list(lin = 1, quad = 2)))
summary(CC.aov, split = list(P = list(lin = 1, quad = 2)),
        expand.split = FALSE)
```

summary.glm

[methods](#)glmsummary.glm

```
## S3 method for class 'glm'
summary(object, dispersion = NULL, correlation = FALSE,
        symbolic.cor = FALSE, ...)

## S3 method for class 'summary.glm'
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"),
      show.residuals = FALSE, ...)
```

```
object      "glm"glm
x            "summary.glm"summary.glm
dispersion  NULLobject
correlation  TRUE
digits
symbolic.cor TRUEsymnum
signif.stars TRUE
show.residuals TRUE
...
```

```
print.summary.glmcoefTRUEcoefficientst ratioz ratioNaN
print
summary(object)$correlation
dispersionNULL1binomialPoisson
summaryglmsummary.lm
```

```
summary.glm"summary.glm"

call          object
family        object
deviance       object
contrasts      object
df.residual    object
null.deviance  object
df.null        object
deviance.resid residuals.glm
coefficients
aliased
dispersion     NULL
df
cov.unscaled   dispersion = 1
cov.scaled     dispersion
correlation     correlation
symbolic.cor   correlationsymbolic.cor
```

[glmsummary](#)

```
## For examples see example(glm)
```

```
summary.lm
```

```
summary"lm"

## S3 method for class 'lm'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)

## S3 method for class 'summary.lm'
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)

object      "lm"lm
x            "summary.lm"summary.lm
correlation  TRUE
digits
symbolic.cor TRUEsymnum
signif.stars TRUE
...
```

```
print.summary.lmsignif.starsTRUE
print
summary(object)$correlation
```

```
summary.lmobject"call""terms"
```

```
residuals      lm
coefficients     $p \times 4$ 
aliases
sigma
```

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_i w_i R_i^2,$$

```
               $R_i$ residuals[i]
df              ( $p, n-p, p^*$ )
fstatistic
r.squared        $R^2$ 
```

$$R^2 = 1 - \frac{\sum_i R_i^2}{\sum_i (y_i - y^*)^2},$$

```
               $y^* y_i$ 
adj.r.squared   $R^2 p$ 
cov.unscaled    $p \times p \hat{\beta}_j \hat{\beta}_j = 1, \dots, p$ 
correlation     cov.unscaledcorrelation = TRUE
symbolic.cor    correlationsymbolic.cor
na.action       object
```

```
lmsummary
```

```
coef
```

```
##-- Continuing the lm(.) example:
coef(lm.D90) # the bare coefficients
sld90 <- summary(lm.D90 <- lm(weight ~ group -1)) # omitting intercept
sld90
coef(sld90) # much more
```

```
## model with *aliased* coefficient:
lm.D9. <- lm(weight ~ group + I(group != "Ctl"))
Sm.D9. <- summary(lm.D9.)
Sm.D9. # shows the NA NA NA NA line
stopifnot(length(cc <- coef(lm.D9.)) == 3, is.na(cc[3]),
           dim(coef(Sm.D9.)) == c(2,4), Sm.D9.$df == c(2, 18, 3))
```

summary.manova

summary"manova"

```
## S3 method for class 'manova'
summary(object,
        test = c("Pillai", "Wilks", "Hotelling-Lawley", "Roy"),
        intercept = FALSE, tol = 1e-7, ...)
```

```
object          "manova" aov
test
intercept       TRUE
tol             qr
...
```

summary.manova

tol

"summary.manova"

row.names stats

SS

Eigenvalues

stats

row.namesSSDf

[manovaov](#)

```
## Example on producing plastic film from Krzanowski (1998, p. 381)
tear <- c(6.5, 6.2, 5.8, 6.5, 6.5, 6.9, 7.2, 6.9, 6.1, 6.3,
          6.7, 6.6, 7.2, 7.1, 6.8, 7.1, 7.0, 7.2, 7.5, 7.6)
gloss <- c(9.5, 9.9, 9.6, 9.6, 9.2, 9.1, 10.0, 9.9, 9.5, 9.4,
           9.1, 9.3, 8.3, 8.4, 8.5, 9.2, 8.8, 9.7, 10.1, 9.2)
opacity <- c(4.4, 6.4, 3.0, 4.1, 0.8, 5.7, 2.0, 3.9, 1.9, 5.7,
             2.8, 4.1, 3.8, 1.6, 3.4, 8.4, 5.2, 6.9, 2.7, 1.9)
Y <- cbind(tear, gloss, opacity)
rate <- gl(2,10, labels = c("Low", "High"))
additive <- gl(2, 5, length = 20, labels = c("Low", "High"))

fit <- manova(Y ~ rate * additive)
summary.aov(fit) # univariate ANOVA tables
summary(fit, test = "Wilks") # ANOVA table of Wilks' lambda
summary(fit) # same F statistics as single-df terms
```

```
summary.nls
```

```
summary"nls"
```

```
## S3 method for class 'nls'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
```

```
## S3 method for class 'summary.nls'
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)
```

```
object      "nls"
x            "summary.nls"summary.nls
correlation  TRUE
digits
symbolic.cor TRUEsymbolic.cor
signif.stars TRUE
...
```

```
print.summary.nls$signif.starsTRUE
summary(object)$correlation
```

```
summary.nlsobject"formula"
```

```
residuals      nls
```

```
coefficients    $p \times 4$ 
```

```
sigma
```

$$\hat{\sigma}^2 = \frac{1}{n-p} \sum_i R_i^2,$$

R_i^2

```
df               $(p, n-p)n$ 
```

```
cov.unscaled    $p \times p$ 
```

```
correlation     cov.unscaledcorrelation = TRUE
```

```
symbolic.cor    correlationsymbolic.cor
```

```
nlssummary
```

```
coef
```

```
summary.princomp
```

```
summary"princomp"
```

```
## S3 method for class 'princomp'
```

```
summary(object, loadings = FALSE, cutoff = 0.1, ...)
```

```
## S3 method for class 'summary.princomp'
```

```
print(x, digits = 3, loadings = x$print.loadings,  
      cutoff = x$cutoff, ...)
```

```
object          "princomp"princomp()
```

```
loadings
```

```
cutoff
```

```
x               "summary.princomp"
```

```
digits
```

```
...
```

```
objectcutoffprint.loadings
```

```
princomp
```

```
summary(pc.cr <- princomp(USArrests, cor = TRUE))
```

```
## The signs of the loading columns are arbitrary
```

```
print(summary(princomp(USArrests, cor = TRUE),  
              loadings = TRUE, cutoff = 0.2), digits = 2)
```

`supsmu`

```
supsmu(x, y, wt =, span = "cv", periodic = FALSE, bass = 0, trace = FALSE)
```

```
x
y
wt
span      "cv"
periodic  TRUE[0, 1]
bass
trace     "cv"
```

```
supsmuk/2k0.5 * n0.2 * n0.05 * nnspan * n
```

```
n < 40span > 0
xywt
```

```
x
y
```

`ppr`

```
require(graphics)
```

```
with(cars, {
  plot(speed, dist)
  lines(supsmu(speed, dist))
  lines(supsmu(speed, dist, bass = 7), lty = 2)
})
```

symnum

```
symnum(x, cutpoints = c(0.3, 0.6, 0.8, 0.9, 0.95),
       symbols = if(numeric.x) c(" ", ".", ",", "+", "*", "B")
       else c(".", "|"),
       legend = length(symbols) >= 3,
       na = "?", eps = 1e-5, numeric.x = is.numeric(x),
       corr = missing(cutpoints) && numeric.x,
       show.max = if(corr) "1", show.min = NULL,
       abbr.colnames = has.colnames,
       lower.triangular = corr && is.numeric(x) && is.matrix(x),
       diag.lower.tri = corr && !is.null(show.max))
```

```
x
cutpoints      cutpoints[j]=  $c_j$ corr
symbols        corrcutpointssymbols[j]=  $s_j(c_j, c_{j+1}]$ 
               numeric.xFALSExlogicalc(".", "|")
legend         "legend"
na             NA  
sna == FALSENA"legend"
eps
numeric.x      x
corr           TRUEx01abs(x)
show.max       TRUEcharacter
show.min       TRUEcharacter
abbr.colnames  NULLNULLFALSEx""abbr.colnamesTRUEabbreviate(*, minlength =
               abbr.colnames)
lower.triangular
               TRUEx
diag.lower.tri lower.triangularTRUE
```

[noquote](#)x

legendTRUE"legend"

$$c_1s_1c_2s_2\ldots s_nc_{n+1}$$

c_j = cutpoints[j] s_j = symbols[j]

<maechler@stat.math.ethz.ch>

as.characterimage

```
ii <- setNames(0:8, 0:8)
symnum(ii, cutpoints = 2*(0:4), symbols = c(".", "-", "+", "$"))
symnum(ii, cutpoints = 2*(0:4), symbols = c(".", "-", "+", "$"), show.max = TRUE)

symnum(1:12 %% 3 == 0) # --> "|" = TRUE, "." = FALSE for logical

## Pascal's Triangle modulo 2 -- odd and even numbers:
N <- 38
pascal <- t(sapply(0:N, function(n) round(choose(n, 0:N - (N-n)%/2))))
rownames(pascal) <- rep("", 1+N) # <-- to improve "graphic"
symnum(pascal %% 2, symbols = c(" ", "A"), numeric.x = FALSE)

##-- Symbolic correlation matrices:
symnum(cor(attitude), diag.lower.tri = FALSE)
symnum(cor(attitude), abbr.colnames = NULL)
symnum(cor(attitude), abbr.colnames = FALSE)
symnum(cor(attitude), abbr.colnames = 2)

symnum(cor(rbind(1, rnorm(25), rnorm(25)^2)))
symnum(cor(matrix(rexp(30, 1), 5, 18))) # <-- PATTERN ! --
symnum(cm1 <- cor(matrix(rnorm(90), 5, 18))) # < White Noise SMALL n
symnum(cm1, diag.lower.tri = FALSE)
symnum(cm2 <- cor(matrix(rnorm(900), 50, 18))) # < White Noise "BIG" n
symnum(cm2, lower.triangular = FALSE)

## NA's:
Cm <- cor(matrix(rnorm(60), 10, 6)); Cm[c(3,6), 2] <- NA
symnum(Cm, show.max = NULL)

## Graphical P-values (aka "significance stars"):
pval <- rev(sort(c(outer(1:6, 10^-(1:3)))))
symp <- symnum(pval, corr = FALSE,
               cutpoints = c(0, .001, .01, .05, .1, 1),
               symbols = c("***", "**", "*", ".", " "))
noquote(cbind(P.val = format(pval), Signif = symp))
```

t.test

t.test(x, ...)

Default S3 method:

```
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
```

```
## S3 method for class 'formula'
t.test(formula, data, subset, na.action = na.pass, ...)
```

```
x
y
alternative      "two.sided""greater""less"
mu
paired
var.equal        TRUE
conf.level
formula          lhs ~ rhslhsrhs1lhs"Pair"rhs1
data              model.frameformulaenvironment(formula)
subset
na.action         NA
...               formulapaired
```

```
alternative = "greater"xy
pairedTRUExpairedTRUEvar.equalTRUEvar.equalFALSE
```

```
t.test()wilcox.test()Infxy
```

```
"htest"
statistic
parameter
p.value
conf.int
estimate
null.value
stderr
alternative
method
data.name
```

```
prop.test
```

```

## Two-sample t-test
t.test(1:10, y = c(7:20))      # P = .00001855
t.test(1:10, y = c(7:20, 200)) # P = .1245    -- NOT significant anymore

## Traditional interface
with(mtcars, t.test(mpg[am == 0], mpg[am == 1]))

## Formula interface
t.test(mpg ~ am, data = mtcars)

## One-sample t-test
## Traditional interface
t.test(sleep$extra)

## Formula interface
t.test(extra ~ 1, data = sleep)

## Paired t-test
## The sleep data is actually paired, so could have been in wide format:
sleep2 <- reshape(sleep, direction = "wide",
                  idvar = "ID", timevar = "group")

## Traditional interface
t.test(sleep2$extra.1, sleep2$extra.2, paired = TRUE)

## Formula interface
t.test(Pair(extra.1, extra.2) ~ 1, data = sleep2)

```

TDist

dfncp

```

dt(x, df, ncp, log = FALSE)
pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
rt(n, df, ncp)

```

xq

p

n length(n) > 1

df > 0df = Inf

ncp δ rt()abs(ncp) <= 37.62

loglog.p

lower.tail $P[X \leq x]P[X > x]$

$$t\mathbf{d}\mathbf{f}=\nu$$

$$f(x)=\frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu}\Gamma(\nu/2)}(1+x^2/\nu)^{-(\nu+1)/2}$$

$$x0\nu>1\frac{\nu}{\nu-2}\nu>2$$

$$t(\nu,\delta)=\left(\mathbf{d}\mathbf{f},\ \mathbf{n}\mathbf{c}\mathbf{p}\right)T_{\nu}(\delta):=(U+\delta)/\sqrt{V/\nu}UVU\sim\mathcal{N}(0,1)V\sim\chi^2_{\nu}$$

$$\begin{array}{l}t\\T=\frac{\bar{X}-\mu_0}{S/\sqrt{n}}\bar{X}\textcolor{blue}{\mathbf{mean}}S\textcolor{blue}{\mathbf{sd}}X_1,X_2,\ldots,X_n\mathcal{N}(\mu,\sigma^2)Tt\mathbf{d}\mathbf{f}=n-1\mathbf{n}\mathbf{c}\mathbf{p}=(\mu-\mu_0)\sqrt{n}/\sigma\end{array}$$

$$tF_{\nu}F_{\nu}(t)=\tfrac{1}{2}I_x(\tfrac{\nu}{2},\tfrac{1}{2}),t\leq 0F_{\nu}(t)=1-\tfrac{1}{2}I_x(\tfrac{\nu}{2},\tfrac{1}{2}),t\geq 0x:=\nu/(\nu+t^2)I_x(a,b)\textcolor{blue}{\mathbf{pbeta}}(\mathbf{x},\ \mathbf{a},\mathbf{b})$$

$$\mathbf{d}\mathbf{t}\mathbf{p}\mathbf{t}\mathbf{q}\mathbf{t}\mathbf{r}\mathbf{t}$$

$$\mathbf{N}\mathbf{a}\mathbf{N}$$

$$\mathbf{n}\mathbf{r}\mathbf{t}$$

$$\mathbf{n}$$

$$\mathbf{n}\mathbf{c}\mathbf{p}=\emptyset\mathbf{n}\mathbf{c}\mathbf{p}\mathbf{n}\mathbf{c}\mathbf{p}$$

$$\mathbf{n}\mathbf{c}\mathbf{p}\mathbf{n}\mathbf{c}\mathbf{p}$$

$$\mathbf{d}\mathbf{t}\mathbf{d}\mathbf{b}\mathbf{i}\mathbf{n}\mathbf{o}\mathbf{m}$$

$$\mathbf{d}\mathbf{t}x\neq 0$$

$$\mathbf{p}\mathbf{t}\textcolor{blue}{\mathbf{pbeta}}$$

$$\mathbf{p}\mathbf{t}$$

$$t$$

$$\mathbf{q}\mathbf{t}$$

$$\textcolor{blue}{\mathbf{d}\mathbf{f}}$$

```

require(graphics)

1 - pt(1:5, df = 1)
qt(.975, df = c(1:10,20,50,100,1000))

tt <- seq(0, 10, length.out = 21)
ncp <- seq(0, 6, length.out = 31)
ptn <- outer(tt, ncp, function(t, d) pt(t, df = 3, ncp = d))
t.tit <- "Non-central t - Probabilities"
image(tt, ncp, ptn, zlim = c(0,1), main = t.tit)
persp(tt, ncp, ptn, zlim = 0:1, r = 2, phi = 20, theta = 200, main = t.tit,
      xlab = "t", ylab = "non-centrality parameter",
      zlab = "Pr(T <= t)")

plot(function(x) dt(x, df = 3, ncp = 2), -3, 11, ylim = c(0, 0.32),
      main = "Non-central t - Density", yaxs = "i")

## Relation between F_t(.) = pt(x, n) and pbeta():
ptBet <- function(t, n) {
  x <- n/(n + t^2)
  r <- pbeta(x, n/2, 1/2) / 2
  pos <- t > 0
  r[pos] <- 1 - pb[pos]
  r
}
x <- seq(-5, 5, by = 1/8)
nu <- 3:10
pt. <- outer(x, nu, pt)
ptB <- outer(x, nu, ptBet)
## matplot(x, pt., type = "l")
stopifnot(all.equal(pt., ptB, tolerance = 1e-15))

```

termplot

```

termplot(model, data = NULL, envir = environment(formula(model)),
  partial.resid = FALSE, rug = FALSE,
  terms = NULL, se = FALSE,
  xlabs = NULL, ylabs = NULL, main = NULL,
  col.term = 2, lwd.term = 1.5,
  col.se = "orange", lty.se = 2, lwd.se = 1,
  col.res = "gray", cex = 1, pch = par("pch"),
  col.smth = "darkred", lty.smth = 2, span.smth = 2/3,
  ask = dev.interactive() && nb.fig < n.tms,
  use.factor.levels = TRUE, smooth = NULL, ylim = "common",
  plot = TRUE, transform.x = FALSE, ...)

```

```

model
data          model
envir         model
partial.resid
rug
terms          NULLpredict(..., type = "terms", terms = *)
se            2×
xlab
ylab
main           TRUENULLFALSE
col.term1wd.term
              lines
col.selty.selwd.se
              se = TRUE
col.rescexpch  partial.resid = TRUEpoints
ask           TRUEpar(ask=.)
use.factor.levels

```

```

smooth          NULLpanel.smooth
lty.smthcol.smthspan.smth
              smooth
ylim           "common""free"
plot           FALSE
transform.x     TRUEsmooth = panel.smooth
...

```

```

modelpredicttype = "terms"glmcoxphsurvreg
partial.resid = TRUEmodelresidualstype = "partial"lmglm
datatermplotna.action=na.exclude

```

```

plot = FALSE

```

```

plot = FALSExysex"constant"

```

```

plot.lmpredict.glm

```



```

require(graphics)

had.splines <- "package:splines" %in% search()
if(!had.splines) rs <- require(splines)
x <- 1:100
z <- factor(rep(LETTERS[1:4], 25))
y <- rnorm(100, sin(x/10)+as.numeric(z))
model <- glm(y ~ ns(x, 6) + z)

par(mfrow = c(2,2)) ## 2 x 2 plots for same model :
termplot(model, main = paste("termplot( ", deparse(model$call)," ...)")
termplot(model, rug = TRUE)
termplot(model, partial.resid = TRUE, se = TRUE, main = TRUE)
termplot(model, partial.resid = TRUE, smooth = panel.smooth, span.smth = 1/4)
if(!had.splines && rs) detach("package:splines")

if(requireNamespace("MASS", quietly = TRUE)) {
hills.lm <- lm(log(time) ~ log(climb)+log(dist), data = MASS::hills)
termplot(hills.lm, partial.resid = TRUE, smooth = panel.smooth,
         terms = "log(dist)", main = "Original")
termplot(hills.lm, transform.x = TRUE,
         partial.resid = TRUE, smooth = panel.smooth,
         terms = "log(dist)", main = "Transformed")
}

```

terms

terms

terms(x, ...)

x

...

```

"aovlist""terms""formula"terms.formulaterms"terms"model.frame
printlabels"terms"?terms.object

```

```

c("terms", "formula"?terms.object

```

```

terms.objectterms.formulamlmformula

```

terms.formula	terms
---------------	-------

[termsmodel.matrix](#)

```
## S3 method for class 'formula'
terms(x, specials = NULL, abb = NULL, data = NULL, neg.out = TRUE,
      keep.order = FALSE, simplify = FALSE, ...,
      allowDotAsName = FALSE)
```

x	formula
specials	termsNULL
abb	
data	..
neg.out	
keep.order	FALSE
simplify	
...	
allowDotAsName	.data.

```
termskeep.order = TRUEattributeterms.object"variables"
```

[termsterms.object](#)

terms.object

[termsformula](#)

[terms.formula](#)(<formula>)

factors

[integer](#)(0)

term.labels

```
variables list
intercept
order term.labels
response variables
offset offsetoffsetvariables
specials specialterms.formulasspecialsvariables
dataClasses .MFclass
predvars makepredictcall

c("terms", "formula")
```

```
formula
specialsaovcoxph
```

```
termsformula
```

```
## use of specials (as used for gam() in packages mgcv and gam)
(tf <- terms(y ~ x + x:z + s(x), specials = "s"))
## Note that the "factors" attribute has variables as row names
## and term labels as column names, both as character vectors.
attr(tf, "specials") # index 's' variable(s)
rownames(attr(tf, "factors"))[attr(tf, "specials")$s]

## we can keep the order by
terms(y ~ x + x:z + s(x), specials = "s", keep.order = TRUE)
```

```
time
```

```
time
cycle
frequencydeltatts
```

```
time(x, ...)
## Default S3 method:
time(x, offset = 0, ts.eps = getOption("ts.eps"), ...)

cycle(x, ...)
frequency(x, ...)
deltat(x, ...)
```

```

x
offset          00.51
ts.eps          time()ts.epsround()
...

```

```

tspxtimecycles
time()round()ts.epsts.eps = 0

```

```

tsstarttspwindow
datesystem.time

```

```

require(graphics)

cycle(presidents)
# a simple series plot
plot(as.vector(time(presidents)), as.vector(presidents), type = "l")

```

toeplitz

```
toeplitz()
```

```
T1 <- toeplitz(col, row)
```

```
T <- toeplitz2(x, nr, nc)
```

```
(nr, nc) $T_{i,j} = x_{i-j+n_c}$  $n_c = \text{ncol}(T)$ 
```

```

toeplitz(x, r = NULL, symmetric = is.null(r))
toeplitz2(x, nrow = length(x) + 1 - ncol, ncol = length(x) + 1 - nrow)

```

```

x          toeplitz(x, *)toeplitz2(x, *)T[i,j] == x[i-j + ncol]
r
symmetric  logical
nrowncol

```

$n \times mT$

```
toeplitz() dim(T)(n,m)m == length(x)n == mn == length(r)
```

```
toeplitz2() dim(T) == c(nrow, ncol)
```

```
x <- 1:5
```

```
toeplitz(x)
```

```
T. <- toeplitz(1:5, 11:13) # with a *Warning* x[1] != r[1]
```

```
T2 <- toeplitz2(c(13:12, 1:5), 5, 3)# this is the same matrix:
```

```
stopifnot(identical(T., T2))
```

```
# Matrix of character (could also have logical, raw, complex ...) {also warning}:  
noquote(toeplitz(letters[1:4], LETTERS[20:26]))
```

```
## A convolution/smoother weight matrix :
```

```
m <- 17
```

```
k <- length(wts <- c(76, 99, 60, 20, 1))
```

```
n <- m-k+1
```

```
## Convolution
```

```
W <- toeplitz2(c(rep(0, m-k), wts, rep(0, m-k)), ncol=n)
```

```
## "display" nicely :
```

```
if(requireNamespace("Matrix"))
```

```
  print(Matrix::Matrix(W)) else {
```

```
  colnames(W) <- paste0(",", if(n <= 9) 1:n else c(1:9, letters[seq_len(n-9)]))
```

```
  print(W)
```

```
}
```

```
## scale W to have column sums 1:
```

```
W. <- W / sum(wts)
```

```
all.equal(rep(1, ncol(W.)), colSums(W.), check.attributes = FALSE)
```

```
## Visualize "mass-preserving" convolution
```

```
x <- 1:n; f <- function(x) exp(-((x - .4*n)/3)^2)
```

```
y <- f(x) + rep_len(3:-2, n)/10
```

```
## Smoothing convolution:
```

```
y.hat <- W. %*% y # y.hat := smoothed(y) ("mass preserving" -> longer than y)
```

```
stopifnot(length(y.hat) == m, m == n + (k-1))
```

```
plot(x,y, type="b", xlim=c(1,m)); curve(f(x), 1,n, col="gray", lty=2, add=TRUE)
```

```
lines(1:m, y.hat, col=2, lwd=3)
```

```
rbind(sum(y), sum(y.hat)) ## mass preserved
```

```
## And, yes, convolve(y, *) does the same when called appropriately:
```

```
all.equal(c(y.hat), convolve(y, rev(wts/sum(wts)), type="open"))
```

ts

ts

as.tsis.ts

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
    deltat = 1, ts.eps = getOption("ts.eps"),
    class = if(nseries > 1) c("mts", "ts", "matrix", "array") else "ts",
    names = )
as.ts(x, ...)
is.ts(x)

is.mts(x)
```

data data.matrix

start

end start

frequency

deltat frequencydeltat

ts.eps ts.epsend - startfrequency

class NULL"none""ts"c("mts", "ts", "matrix", "array")

names data"Series 1""Series 2"

x

...

ts"ts"data

"ts"EuStockMarkets[, "DAX"][window](#)t"ts"

frequency17frequency12412printfrequencyfrequency = 0.2

as.ts[tsp](#)

is.ts()"ts"

is.mts(x)xis.ts(x)is.matrix(x)"mts"

[tsp](#)[frequency](#)[start](#)[end](#)[time](#)[window](#)[print](#).[tsplot](#).[ts](#)

<https://CRAN.R-project.org/view=TimeSeries>

```

require(graphics)

ts(1:10, frequency = 4, start = c(1959, 2)) # 2nd Quarter of 1959
print( ts(1:10, frequency = 7, start = c(12, 2)), calendar = TRUE)
# print.ts(.)
## Using July 1954 as start date:
gnp <- ts(cumsum(1 + round(rnorm(100), 2)),
          start = c(1954, 7), frequency = 12)
plot(gnp) # using 'plot.ts' for time-series plot

utils:: methods(class = "ts") # all functions with methods available for "ts"

## "ts" methods for head() and tail()
utils:: head(lynx, 4)
utils:: tail(lynx, -7)

## Multivariate
z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
class(z)
is.mts(z)
head(z) # "ts" method ==> incl. times:
plot(z)
plot(z, plot.type = "single", lty = 1:3)

## A phase plot:
plot(nhtemp, lag(nhtemp, 1), cex = .8, col = "blue",
     main = "Lag plot of New Haven temperatures")

```

ts-methods

"ts"[ts](#)

```
## S3 method for class 'ts'
diff(x, lag = 1, differences = 1, ...)
```

```
## S3 method for class 'ts'
na.omit(object, ...)
```

```

x                "ts"
lag
differences
object
...

```

```
na.omit
```

na.omitobject

diffna.omitna.failna.contiguous

ts.plot

plot.ts

ts.plot(..., gpars = list())

...
gpars ...

plot

plot.ts

require(graphics)
ts.plot(ldeaths, mdeaths, fdeaths,
 gpars=list(xlab="year", ylab="deaths", lty=c(1:3)))

ts.union

ts.unionNAts.intersect

ts.intersect(..., dframe = FALSE)
ts.union(..., dframe = FALSE)

...
dframe TRUE

...

dframeFALSE

[cbind](#)

```
ts.union(mdeaths, fdeaths)
cbind(mdeaths, fdeaths) # same as the previous line
ts.intersect(window(mdeaths, 1976), window(fdeaths, 1974, 1978))

sales1 <- ts.union(BJsales, lead = BJsales.lead)
ts.intersect(sales1, lead3 = lag(BJsales.lead, -3))
```

[tsdiag](#)

`tsdiag(object, gof.lag, ...)`

object
gof.lag
...

gof.lag
[arimaStructTS](#)

[arimaStructTSBox.test](#)

```
require(graphics)

fit <- arima(lh, c(1,0,0))
tsdiag(fit)

## see also examples(arima)

(fit <- StructTS(log10(JohnsonJohnson), type = "BSM"))
tsdiag(fit)
```

tsp

tsptspNULLtsp<-tsphasTspxtsp

tsp(x)
tsp(x) <- value
hasTsp(x)

x
value NULL

tspts

NULLtsp"ts""mts"x

xtspNULL
hasTspNROW(x)

tstart

tsSmooth

tsSmooth(object, ...)

object "StructTS"
...

[KalmanSmoothStructTS](#)
[AirPassengersJohnsonJohnsonNile](#)

Tukey

$$R/sRdf \times s^2df$$

[pchisq](#)

```
ptukey(q, nmeans, df, nranges = 1, lower.tail = TRUE, log.p = FALSE)
qtukey(p, nmeans, df, nranges = 1, lower.tail = TRUE, log.p = FALSE)
```

```
q
p
nmeans
df          s
nranges
log.p
lower.tail   $P[X \leq x]P[X > x]$ 
```

$n_g = \text{nranges} R n_g \text{nmeans}$

ptukeyqtukey

ptukeyqtukeyptukeyqtukey

qtukey

[pnormqnorm](#)

```
if(interactive())
  curve(ptukey(x, nm = 6, df = 5), from = -1, to = 8, n = 101)
(ptt <- ptukey(0:10, 2, df = 5))
(qtt <- qtukey(.95, 2, df = 2:11))
## The precision may be not much more than about 8 digits:
summary(abs(.95 - ptukey(qtt, 2, df = 2:11)))
```

TukeyHSD

```
TukeyHSD(x, which, ordered = FALSE, conf.level = 0.95, ...)
```

```
x          aov
which
ordered    orderedlwr
conf.level
...
```

```
"aov"
```

```
which
```

```
c("multicomp", "TukeyHSD")whichdiffLwruprj adj
printplot"TukeyHSD"plotxlabylabmain
```

```
aovqtukeymodel.tablesglht
```

```
require(graphics)
```

```
summary(fm1 <- aov(breaks ~ wool + tension, data = warpbreaks))
TukeyHSD(fm1, "tension", ordered = TRUE)
plot(TukeyHSD(fm1, "tension"))
```

Uniform

minmaxdunifpunifqunifrunif

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

```
xq
p
n          length(n) > 1
minmax
loglog.p
lower.tail   $P[X \leq x]P[X > x]$ 
```

minmax01

$$f(x) = \frac{1}{\max - \min}$$

$\min \leq x \leq \max$

$u := \min == \max X \equiv u$ dunifNaN

runifmax = minmax-minmin

dunifpunifqunifrunif

nrunif

n

[.Random.seed](#)

[RNG](#)

```
u <- runif(20)
```

```
## The following relations always hold :
```

```
punif(u) == u
```

```
dunif(u) == 1
```

```
var(runif(10000)) #- ~ = 1/12 = .08333
```

uniroot

unirootlowerupperf

extendInt="no"interval = c(lower,upper)sign(f(x))

```
uniroot(f, interval, ...,
        lower = min(interval), upper = max(interval),
        f.lower = f(lower, ...), f.upper = f(upper, ...),
        extendInt = c("no", "yes", "downX", "upX"), check.conv = FALSE,
        tol = .Machine$double.eps^0.25, maxiter = 1000, trace = 0)
```

f

interval

... f

lowerupper

f.lowerf.upper f(upper)f(lower)f()

extendInt c(lower,upper)f()"no"

check.conv maxiter

tol

maxiter

trace

...

intervallowerupperextendInt="no"extendInt="yes" $[l, u]f(l) \cdot f(u) \leq 0$

$f(x_0)$ extendInt"upX""downX" $S := \pm 1S = \text{sign}(f(x_0 + \epsilon))[l, u]S \cdot f(l) \leq 0S \cdot f(u) \geq 0$

uniroot()zero

f(x) == 0xtolx

maxiter

ff(, ...)

f

rootf.rootiterestim.precrootNAinit.itextendIntNAextendIntitersumzero

zero.<https://netlib.org/c/brent.shar>

polyrootoptimizenlm

```
require(utils) # for str

## some platforms hit zero exactly on the first step:
## if so the estimated precision is 2/3.
f <- function(x, a) x - a
str(xmin <- uniroot(f, c(0, 1), tol = 0.0001, a = 1/3))

## handheld calculator example: fixed point of cos(.):
uniroot(function(x) cos(x) - x, lower = -pi, upper = pi, tol = 1e-9)$root

str(uniroot(function(x) x*(x^2-1) + .5, lower = -2, upper = 2,
             tol = 0.0001))
str(uniroot(function(x) x*(x^2-1) + .5, lower = -2, upper = 2,
             tol = 1e-10))

## Find the smallest value x for which exp(x) > 0 (numerically):
r <- uniroot(function(x) 1e80*exp(x) - 1e-300, c(-1000, 0), tol = 1e-15)
str(r, digits.d = 15) # around -745, depending on the platform.

exp(r$root)      # = 0, but not for r$root * 0.999...
minexp <- r$root * (1 - 10*.Machine$double.eps)
exp(minexp)      # typically denormalized

##--- uniroot() with new interval extension + checking features: -----

f1 <- function(x) (121 - x^2)/(x^2+1)
f2 <- function(x) exp(-x)*(x - 12)

try(uniroot(f1, c(0,10)))
try(uniroot(f2, c(0, 2)))
##--> error: f() .. end points not of opposite sign

## where as 'extendInt="yes"' simply first enlarges the search interval:
u1 <- uniroot(f1, c(0,10),extendInt="yes", trace=1)
u2 <- uniroot(f2, c(0,2), extendInt="yes", trace=2)
stopifnot(all.equal(u1$root, 11, tolerance = 1e-5),
          all.equal(u2$root, 12, tolerance = 6e-6))

## The *danger* of interval extension:
## No way to find a zero of a positive function, but
## numerically, f(-|M|) becomes zero :
u3 <- uniroot(exp, c(0,2), extendInt="yes", trace=TRUE)

## Nonsense example (must give an error):
tools::assertCondition( uniroot(function(x) 1, 0:1, extendInt="yes"),
                        "error", verbose=TRUE)

## Convergence checking :
sinc <- function(x) ifelse(x == 0, 1, sin(x)/x)
curve(sinc, -6,18); abline(h=0,v=0, lty=3, col=adjustcolor("gray", 0.8))
```

```

uniroot(sinc, c(0,5), extendInt="yes", maxiter=4) #-> "just" a warning

## now with check.conv=TRUE, must signal a convergence error :

uniroot(sinc, c(0,5), extendInt="yes", maxiter=4, check.conv=TRUE)

### Weibull cumulative hazard (example origin, Ravi Varadhan):
cumhaz <- function(t, a, b) b * (t/b)^a
froot <- function(x, u, a, b) cumhaz(x, a, b) - u

n <- 1000
u <- -log(runif(n))
a <- 1/2
b <- 1
## Find failure times
ru <- sapply(u, function(x)
  uniroot(froot, u=x, a=a, b=b, interval= c(1.e-14, 1e04),
    extendInt="yes")$root)
ru2 <- sapply(u, function(x)
  uniroot(froot, u=x, a=a, b=b, interval= c(0.01, 10),
    extendInt="yes")$root)
stopifnot(all.equal(ru, ru2, tolerance = 6e-6))

r1 <- uniroot(froot, u= 0.99, a=a, b=b, interval= c(0.01, 10),
  extendInt="up")
stopifnot(all.equal(0.99, cumhaz(r1$root, a=a, b=b)))

## An error if 'extendInt' assumes "wrong zero-crossing direction":

uniroot(froot, u= 0.99, a=a, b=b, interval= c(0.1, 10), extendInt="down")

```

update

```

updateupdate
update()getCall()x$call
update()getCall()

```

```

update(object, ...)
## Default S3 method:
update(object, formula., ..., evaluate = TRUE)

getCall(x, ...)

```

```

objectx      lmgln
formula.     update.formula
...          name = NULLname
evaluate

```



```
evaluate = TRUE
```

update.formula

```
oldcon <- options(contrasts = c("contr.treatment", "contr.poly"))
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D9
summary(lm.D9)
update(lm.D9, . ~ . - 1)
options(contrasts = c("contr.helmert", "contr.poly"))
update(lm.D9)
getCall(lm.D9) # "through the origin"

options(oldcon)
```

update.formula

update.formula

```
## S3 method for class 'formula'
update(old, new, ...)
```

```
old
new
...
```

oldnewas.formula

```
oldnewoldnewoldnewterms.formula(simplify = TRUE)
```

```
old
```

termsmodel.matrix

```
update(y ~ x,      ~ . + x2) #> y ~ x + x2
update(y ~ x, log(.) ~ .) #> log(y) ~ x
update(. ~ u+v, res ~ .) #> res ~ u + v
```

var.test

```
var.test(x, ...)  
  
## Default S3 method:  
var.test(x, y, ratio = 1,  
         alternative = c("two.sided", "less", "greater"),  
         conf.level = 0.95, ...)  
  
## S3 method for class 'formula'  
var.test(formula, data, subset, na.action, ...)  
  
xy          "lm"  
ratio       xy  
alternative  "two.sided""greater""less"  
conf.level  
formula     lhs ~ rhslhsrhs  
data        model.frameformulaenvironment(formula)  
subset  
na.action   NAgetOption("na.action")  
...  
  
xyxyratio  
  
"htest"  
statistic  
parameter  
p.value  
conf.int  
estimate    xy  
null.value  
alternative  
method      "F test to compare two variances"  
data.name  
  
bartlett.testansari.testmood.test  
  
x <- rnorm(50, mean = 0, sd = 2)  
y <- rnorm(30, mean = 1, sd = 1)  
var.test(x, y) # Do x and y have the same variance?  
var.test(lm(x ~ 1), lm(y ~ 1)) # The same.
```

varimax

```
varimax(x, normalize = TRUE, eps = 1e-5)
promax(x, m = 4)
```

```
x           $pk < p$ 
m          promax
normalize   x
eps
```

```
x %%% Tvarimaxpromax
```

```
loadings    x %%% rotmat"loadings"
rotmat
```

factanalHarman74.com

```
## varimax with normalize = TRUE is the default
fa <- factanal( ~., 2, data = swiss)
varimax(loadings(fa), normalize = FALSE)
promax(loadings(fa))
```

vcov

coefsigma

```
vcov(object, ...)
## S3 method for class 'lm'
vcov(object, complete = TRUE, ...)
## and also for '[summary.]glm' and 'mlm'
## S3 method for class 'aov'
vcov(object, complete = FALSE, ...)

.vcov.aliased.aliased, vc, complete = TRUE)
```

```
object      summary()
complete    aovlmglmmlsummary.lmcoef(.)NAcomplete = TRUEvcov()coef()
...         glmdispersion
aliased     logicalis.na(coef(.))
vc
```

```
vcov()vcov.lmmlglmmlsummary.lmsummary.glmnegbinpolrrlmmultinomglslmecoxph
survreg
vcov()summary(mod)vcov(mod)
.vcov.aliased()vcovvcNAaliasedvclength.aliased)
```

coef

```
NAaliasvcov()complete = TRUElmaovNAcoef()NA
```

Weibull

shapescale

```
dweibull(x, shape, scale = 1, log = FALSE)
pweibull(q, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
qweibull(p, shape, scale = 1, lower.tail = TRUE, log.p = FALSE)
rweibull(n, shape, scale = 1)
```

xq
 p
 n length(n) > 1
 shapescal
 loglog.p
 lower.tail $P[X \leq x]P[X > x]$

shapeascale σ

$$f(x) = (a/\sigma)(x/\sigma)^{a-1} \exp(-(x/\sigma)^a)$$

$$x > 0 F(x) = 1 - \exp(-(x/\sigma)^a) x > 0 E(X) = \sigma \Gamma(1 + 1/a) Var(X) = \sigma^2 (\Gamma(1 + 2/a) - (\Gamma(1 + 1/a))^2)$$

dweibullpweibullqweibullrweibull
 NaN
 nrweibull
 n

$H(t) = -\log(1 - F(t))$
 -pweibull(t, a, b, lower = FALSE, log = TRUE)
 $H(t) = (t/b)^a$

[dpq]weibullrweibull

```
x <- c(0, rlnorm(50))
all.equal(dweibull(x, shape = 1), dexp(x))
all.equal(pweibull(x, shape = 1, scale = pi), pexp(x, rate = 1/pi))
## Cumulative hazard H():
all.equal(pweibull(x, 2.5, pi, lower.tail = FALSE, log.p = TRUE),
          -(x/pi)^2.5, tolerance = 1e-15)
all.equal(qweibull(x/11, shape = 1, scale = pi), qexp(x/11, rate = 1/pi))
```

`weighted.mean`

```
weighted.mean(x, w, ...)

## Default S3 method:
weighted.mean(x, w, ..., na.rm = FALSE)
```

```
x
w      xx
...
na.rm  NAx
```

```
x"POSIXct""POSIXlt""difftime""Date"[sum
wxNaN
wx
```

`mean`

```
## GPA from Siegel 1994
wt <- c(5, 5, 4, 1)/15
x <- c(3.7, 3.3, 3.5, 2.8)
xm <- weighted.mean(x, wt)
```

`weighted.residuals`

```
weighted.residuals(obj, drop0 = TRUE)
```

```
obj      lmg1m
drop0    TRUEweights == 0
```

`lm` $R_i\sqrt{w_i}w_i$ `weights``lm`
`influence`

`n'``n'``drop0` = TRUE

`residuals``lm.influence`

```
## following on from example(lm)

all.equal(weighted.residuals(lm.D9),
          residuals(lm.D9))
x <- 1:10
w <- 0:9
y <- rnorm(x)
weighted.residuals(lmxy <- lm(y ~ x, weights = w))
weighted.residuals(lmxy, drop0 = FALSE)
```

`weights`

`weights`
`na``predict`

`weights(object, ...)`

`object`
...

`object`"`weights`"NULL`na``predict`

`weights.glm`

`wilcox.test`

```
wilcox.test(x, ...)  
  
## Default S3 method:  
wilcox.test(x, y = NULL,  
            alternative = c("two.sided", "less", "greater"),  
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,  
            conf.int = FALSE, conf.level = 0.95,  
            tol.root = 1e-4, digits.rank = Inf, ...)  
  
## S3 method for class 'formula'  
wilcox.test(formula, data, subset, na.action = na.pass, ...)  
  
x  
y  
alternative      "two.sided""greater""less"  
mu  
paired  
exact  
correct  
conf.int  
conf.level  
tol.root        conf.intuniroot(*, tol=tol.root)  
digits.rank     rank(signif(r, digits.rank))rank(r)  
formula         lhs ~ rhslhsrhs1lhs"Pair"rhs1  
data            model.frameformulaenvironment(formula)  
subset  
na.action       NA  
...            formulapaired  
  
xxypairedTRUExx - ymu  
xypairedFALSExymu"greater"xy  
exact  
digits.rank = 7  
conf.intxyF(u + v)/2wvFFxy  
alternative  
  
xyconf.int = TRUEestimateNaN
```



```

"htest"
statistic
parameter
p.value
null.value      mu
alternative
method
data.name
conf.int        conf.int = TRUE
estimate        conf.int = TRUE

```

```
exact = TRUE
```

```

 $m(m+1)/2m$ 
(x[i], y[j])y[j]x[i]

```

```

psignrankwilcox
wilcox_test
kruskal.testtt.test

```

```

require(graphics)
## One-sample test.
## Hollander & Wolfe (1973), 29f.
## Hamilton depression scale factor measurements in 9 patients with
## mixed anxiety and depression, taken at the first (x) and second
## (y) visit after initiation of a therapy (administration of a
## tranquilizer).
x <- c(1.83, 0.50, 1.62, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
wilcox.test(x, y, paired = TRUE, alternative = "greater")
wilcox.test(y - x, alternative = "less") # The same.
wilcox.test(y - x, alternative = "less",
            exact = FALSE, correct = FALSE) # H&W large sample
                                           # approximation

## Formula interface to one-sample and paired tests

depression <- data.frame(first = x, second = y, change = y - x)

```

```

wilcox.test(change ~ 1, data = depression)
wilcox.test(Pair(first, second) ~ 1, data = depression)

## Two-sample test.
## Hollander & Wolfe (1973), 69f.
## Permeability constants of the human chorioamnion (a placental
## membrane) at term (x) and between 12 to 26 weeks gestational
## age (y). The alternative of interest is greater permeability
## of the human chorioamnion for the term pregnancy.
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
wilcox.test(x, y, alternative = "g")          # greater
wilcox.test(x, y, alternative = "greater",
             exact = FALSE, correct = FALSE) # H&W large sample
                                           # approximation

wilcox.test(rnorm(10), rnorm(10, 2), conf.int = TRUE)

## Formula interface.
boxplot(Ozone ~ Month, data = airquality)
wilcox.test(Ozone ~ Month, data = airquality,
             subset = Month %in% c(5, 8))

## accuracy in ties determination via 'digits.rank':
wilcox.test(4:2, 3:1, paired=TRUE) # Warning: cannot compute exact p-value with ties
wilcox.test((4:2)/10, (3:1)/10, paired=TRUE) # no ties => *no* warning
wilcox.test((4:2)/10, (3:1)/10, paired=TRUE, digits.rank = 9) # same ties as (4:2, 3:1)

```

Wilcoxon

mn

```

dwilcox(x, m, n, log = FALSE)
pwilcox(q, m, n, lower.tail = TRUE, log.p = FALSE)
qwilcox(p, m, n, lower.tail = TRUE, log.p = FALSE)
rwilcox(nn, m, n)

```

xq

p

nn length(nn) > 1

mn

loglog.p

lower.tail $P[X \leq x]P[X > x]$

$xymn(x[i], y[j])y[j]x[i]0m * nm * n / 2m * n * (m + n + 1) / 12$

```

dwilcoxpwilcoxqwilcoxrwilcox
nnrwilcox
nn

```

```
wilcox.test
```

```

cwilcox(k, m, n)kmndwilcoxpwilcoxqwilcoxqwilcox
rwilcoxsample().Random.seedRNGkind(sample.kind = ..)

```

```
wilcox.test
```

```
dsignrank
```

```

require(graphics)

x <- -1:(4*6 + 1)
fx <- dwilcox(x, 4, 6)
Fx <- pwilcox(x, 4, 6)

layout(rbind(1,2), widths = 1, heights = c(3,2))
plot(x, fx, type = "h", col = "violet",
      main = "Probabilities (density) of Wilcoxon-Statist.(n=6, m=4)")
plot(x, Fx, type = "s", col = "blue",
      main = "Distribution of Wilcoxon-Statist.(n=6, m=4)")
abline(h = 0:1, col = "gray20", lty = 2)
layout(1) # set back

N <- 200
hist(U <- rwilcox(N, m = 4, n = 6), breaks = 0:25 - 1/2,
     border = "red", col = "pink", sub = paste("N =", N))
mtext("N * f(x), f() = true \"density\"", side = 3, col = "blue")
lines(x, N*fx, type = "h", col = "blue", lwd = 2)
points(x, N*fx, cex = 2)

## Better is a Quantile-Quantile Plot
qqplot(U, qw <- qwilcox((1:N - 1/2)/N, m = 4, n = 6),
       main = paste("Q-Q-Plot of empirical and theoretical quantiles",
                    "Wilcoxon Statistic, (m=4, n=6)", sep = "\n"))
n <- as.numeric(names(print(tU <- table(U))))
text(n+.2, n+.5, labels = tU, col = "red")

```

window

windowxstartend

```
window(x, ...)
## S3 method for class 'ts'
window(x, ...)
## Default S3 method:
window(x, start = NULL, end = NULL,
       frequency = NULL, deltat = NULL, extend = FALSE, ts.eps = getOption("ts.eps"), ...)

window(x, ...) <- value
## S3 replacement method for class 'ts'
window(x, start, end, frequency, deltat, ...) <- value
```

x

start

end

frequencydeltat

extend startend

ts.eps ts.epsstartendts.eps/frequency(x)

...

value

tsstartendstartend

ts

window.default**ts**p

window.tswindow.defaultts

extend = TRUENA

timets

```

window(presidents, 1960, c(1969,4)) # values in the 1960's
window(presidents, deltat = 1) # All Qtr1s
window(presidents, start = c(1945,3), deltat = 1) # All Qtr3s
window(presidents, 1944, c(1979,2), extend = TRUE)

pres <- window(presidents, 1945, c(1949,4)) # values in the 1940's
window(pres, 1945.25, 1945.50) <- c(60, 70)
window(pres, 1944, 1944.75) <- 0 # will generate a warning
window(pres, c(1945,4), c(1949,4), frequency = 1) <- 85:89
pres

```

xtabs

```

xtabs(formula = ~., data = parent.frame(), subset, sparse = FALSE,
      na.action, na.rm = FALSE, addNA = FALSE,
      exclude = if(!addNA) c(NA, NaN), drop.unused.levels = FALSE)

```

```

## S3 method for class 'xtabs'
print(x, na.print = "", ...)

```

```

formula      +
data         model.frameformulaenvironment(formula)
subset
sparse       sparseMatrix
na.action    functionformulasubsetNAna.passna.rmaddNAformulana.omit
na.rm        formulasum
addNA        NAaddNA(*, ifany=TRUE)na.action = na.omit
exclude
drop.unused.levels
              FALSE
x             "xtabs"
na.print      NULLNA""NAa.print = "NA"
...

```

```

summarytablextabs(*, sparse = FALSE)chisq.test
formula
formulaexclude
na.action = na.pass0NA
!addNAa.actionna.omit

```

```
sparse = FALSEc("xtabs", "table")"call"
sparse = TRUEdgTMatrix
```

```
tableas.data.frame.tablextabsDF
sparseMatrix
```

```
## 'esoph' has the frequencies of cases and controls for all levels of
## the variables 'agegp', 'alcgp', and 'tobgp'.
xtabs(cbind(ncases, ncontrols) ~ ., data = esoph)
## Output is not really helpful ... flat tables are better:
ftable(xtabs(cbind(ncases, ncontrols) ~ ., data = esoph))
## In particular if we have fewer factors ...
ftable(xtabs(cbind(ncases, ncontrols) ~ agegp, data = esoph))

## This is already a contingency table in array form.
DF <- as.data.frame(UCBAdmissions)
## Now 'DF' is a data frame with a grid of the factors and the counts
## in variable 'Freq'.
DF
## Nice for taking margins ...
xtabs(Freq ~ Gender + Admit, DF)
## And for testing independence ...
summary(xtabs(Freq ~ ., DF))

## with NA's
DN <- DF; DN[cbind(6:9, c(1:2,4,1))] <- NA
DN # 'Freq' is missing only for (Rejected, Female, B)
(xtNA <- xtabs(Freq ~ Gender + Admit, DN)) # NA prints 'invisibly'
print(xtNA, na.print = "NA") # show NA's better
xtabs(Freq ~ Gender + Admit, DN, na.rm = TRUE) # ignore missing Freq
## Use addNA = TRUE to tabulate missing factor levels:
xtabs(Freq ~ Gender + Admit, DN, addNA = TRUE)
xtabs(Freq ~ Gender + Admit, DN, addNA = TRUE, na.rm = TRUE)
## na.action = na.omit removes all rows with NAs right from the start:
xtabs(Freq ~ Gender + Admit, DN, na.action = na.omit)

## Create a nice display for the warp break data.
warpbreaks$replicate <- rep_len(1:9, 54)
ftable(xtabs(breaks ~ wool + tension + replicate, data = warpbreaks))

### ---- Sparse Examples ----

if(require("Matrix")) withAutoprint({
  ## similar to "nlme"s 'ergoStool' :
  d.ergo <- data.frame(Type = paste0("T", rep(1:4, 9*4)),
    Subj = gl(9, 4, 36*4))
  xtabs(~ Type + Subj, data = d.ergo) # 4 replicates each
  set.seed(15) # a subset of cases:
  xtabs(~ Type + Subj, data = d.ergo[sample(36, 10), ], sparse = TRUE)

  ## Hypothetical two-level setup:
  inner <- factor(sample(letters[1:25], 100, replace = TRUE))
```

```
inout <- factor(sample(LETTERS[1:5], 25, replace = TRUE))
fr <- data.frame(inner = inner, outer = inout[as.integer(inner)])
xtabs(~ inner + outer, fr, sparse = TRUE)
})
```

stats4

stats4-package

`mle()` "mle" logLikAIC

<R-core@r-project.org>

coef-methods	coef
--------------	------

"mle"

signature(object = "ANY") `coef`
signature(object = "mle")
signature(object = "summary.mle")

confint-methods	confint
-----------------	---------

signature(object = "ANY") `confint`
signature(object = "mle")
signature(object = "profile.mle")

logLik-methods	logLik
<p>"mle"</p> <p>signature(object = "ANY") logLik</p> <p>signature(object = "mle")</p> <p>mlenoBIC</p>	
<p>mle</p>	

```
mle(minuslogl, start,
    optim = stats::optim,
    method = if(!useLim) "BFGS" else "L-BFGS-B",
    fixed = list(), nobs, lower, upper, ...)
```

```
minuslogl
start      minuslogl
optim
method     optim
fixed
nobs       BIC
lowerupper optim
...        optim
```

```
optimoptim
minusloglminusloglminuslogl
startfixedupperlowerNAfixedstart+Inf, -Infupperlower
```

```
mle-class
```

```
mll
```

mle-class

```
## Avoid printing to unwarranted accuracy
od <- options(digits = 5)

## Simulated EC50 experiment with count data
x <- 0:10
y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)

## Easy one-dimensional MLE:
nLL <- function(lambda) -sum(stats::dpois(y, lambda, log = TRUE))
fit0 <- mle(nLL, start = list(lambda = 5), nobs = NROW(y))

## sanity check --- notice that "nobs" must be input
## (not guaranteed to be meaningful for any likelihood)
stopifnot(nobs(fit0) == length(y))

# For 1D, this is preferable:
fit1 <- mle(nLL, start = list(lambda = 5), nobs = NROW(y),
           method = "Brent", lower = 1, upper = 20)

## This needs a constrained parameter space: most methods will accept NA
ll <- function(ymax = 15, xhalf = 6) {
  if(ymax > 0 && xhalf > 0)
    -sum(stats::dpois(y, lambda = ymax/(1+x/xhalf), log = TRUE))
  else NA
}
(fit <- mle(ll, nobs = length(y)))
mle(ll, fixed = list(xhalf = 6))

## Alternative using bounds on optimization
ll2 <- function(ymax = 15, xhalf = 6)
  -sum(stats::dpois(y, lambda = ymax/(1+x/xhalf), log = TRUE))
mle(ll2, lower = rep(0, 2))

AIC(fit)
BIC(fit)

summary(fit)
logLik(fit)
vcov(fit)
plot(profile(fit), absVal = FALSE)
confint(fit)

## Use bounded optimization
## The lower bounds are really > 0,
## but we use >=0 to stress-test profiling
(fit2 <- mle(ll2, lower = c(0, 0)))
plot(profile(fit2), absVal = FALSE)

## A better parametrization:
ll3 <- function(lymax = log(15), lxhalf = log(6))
  -sum(stats::dpois(y, lambda = exp(lymax)/(1+exp(lxhalf)), log = TRUE))
```

```

(fit3 <- mle(ll3))
plot(profile(fit3), absVal = FALSE)
exp(confint(fit3))

# Regression tests for bounded cases (this was broken in R 3.x)
fit4 <- mle(ll, lower = c(0, 4)) # has max on boundary
confint(fit4)

## direct check that fixed= and constraints work together
mle(ll, lower = c(0, 4), fixed=list(ymax=23)) # has max on boundary

## Linear regression using MLE
x <- 1:10
y <- c(0.48, 2.24, 2.22, 5.15, 4.64, 5.53, 7, 8.8, 7.67, 9.23)

LM_mll <- function(formula, data = environment(formula))
{
  y <- model.response(model.frame(formula, data))
  X <- model.matrix(formula, data)
  b0 <- numeric(NCOL(X))
  names(b0) <- colnames(X)
  function(b=b0, sigma=1)
    -sum(dnorm(y, X %*% b, sigma, log=TRUE))
}

mll <- LM_mll(y ~ x)

summary(lm(y~x)) # for comparison -- notice variance bias in MLE
summary(mle(mll, lower=c(-Inf,-Inf, 0.01)))
summary(mle(mll, lower=list(sigma = 0.01))) # alternative specification

confint(mle(mll, lower=list(sigma = 0.01)))
plot(profile(mle(mll, lower=list(sigma = 0.01))))

Binom_mll <- function(x, n)
{
  force(x); force(n) ## beware lazy evaluation
  function(p=.5) -dbinom(x, n, p, log=TRUE)
}

## Likelihood functions for different x.
## This code goes wrong, if force(x) is not used in Binom_mll:

curve(Binom_mll(0, 10)(p), xname="p", ylim=c(0, 10))
mll_list <- list(10)
for (x in 1:10)
  mll_list[[x]] <- Binom_mll(x, 10)
for (mll in mll_list)
  curve(mll(p), xname="p", add=TRUE)

mll <- Binom_mll(4,10)
mle(mll, lower = 1e-16, upper = 1-1e-16) # limits must be inside (0,1)

## Boundary case: This works, but fails if limits are set closer to 0 and 1
mll <- Binom_mll(0, 10)
mle(mll, lower=.005, upper=.995)

```

```
## Not run:
## We can use limits closer to the boundaries if we use the
## drop-in replacement optimr() from the optimx package.

mle(mll, lower = 1e-16, upper = 1-1e-16, optim=optimx::optimr)

## End(Not run)

options(od)
```

mle-class	"mle"
-----------	-------

```
new("mle", ...)mle
```

```
call "language"mle
coef "numeric"
fullcoef "numeric"
fixed "numeric"NA
vcov "matrix"
min "numeric"
details "list"optim
minuslogl "function"
nobs "integer"NA
method "character"
```

```
signature(object = "mle")
signature(object = "mle")
signature(fitted = "mle")
signature(object = "mle")nobs
signature(object = "mle")
signature(object = "mle")
signature(object = "mle")
signature(object = "mle")
```

plot-methods	plot
<pre> "mle" ## S4 method for signature 'profile.mle,missing' plot(x, levels, conf = c(99, 95, 90, 80, 50)/100, nseg = 50, absVal = TRUE, ...) x "profile.mle" levels conflevels conf nseg absVal TRUE ... plot signature(x = "ANY", y = "ANY") plot signature(x = "profile.mle", y = "missing") x </pre>	
profile-methods	profile

```

"mle"

## S4 method for signature 'mle'
profile(fitted, which = 1:p, maxsteps = 100, alpha = 0.01,
       zmax = sqrt(qchisq(1 - alpha, 1L)), del = zmax/5,
       trace = FALSE, ...)

fitted
which
maxsteps      zmax
alpha         zmaxzmax
zmax
del
trace
...

```

"profile.mle""profile.mle-class"

signature(fitted = "ANY") [profile](#)
signature(fitted = "mle") "mle"

profile.mle-class	"profile.mle""mle"
-------------------	--------------------

new("profile.mle", ...)profile"mle"

profile "list"zpar.vals.
summary "summary.mle"

signature(object = "profile.mle")
signature(x = "profile.mle", y = "missing")

[mle](#)[mle-class](#)[summary.mle-class](#)

show-methods	show
--------------	------

mlesummary.mle

signature(object = "mle") mle
signature(object = "summary.mle") $-2\log L$

summary-methods	summary
-----------------	---------

signature(object = "ANY")
signature(object = "mle") "summary.mle" $-2\log L$

summary.mle-class	"summary.mle""mle"
-------------------	--------------------

"mle"

new("summary.mle", ...)summary"mle"show

call "language""mle"

coef "matrix"

m2logL "numeric"

signature(object = "summary.mle")object

signature(object = "summary.mle")coef

[summarymlemle-class](#)

update-methods	update
----------------	--------

"mle"

S4 method for signature 'mle'
update(object, ..., evaluate = TRUE)

object

... name = NULLname

evaluate

signature(object = "ANY") [update](#)

signature(object = "mle")

x <- 0:10

y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)

ll <- function(ymax = 15, xhalf = 6)

-sum(stats::dpois(y, lambda = ymax/(1+x/xhalf), log = TRUE))

fit <- mle(ll)

note the recorded call contains ..1, a problem with S4 dispatch

update(fit, fixed = list(xhalf = 3))

vcov-methods

vcov

"mle"

signature(object = "ANY") [vcov](#)

signature(object = "mle")

tcltk

tcltk-package

ls("package:tcltk")
DISPLAY

<R-core@r-project.org>

TclInterface

tclVartclObj

.Tcl(...)
.Tcl.objv(objv)
.Tcl.args(...)
.Tcl.args.objv(...)
.Tcl.callback(...)
.Tk.ID(win)
.Tk.newwin(ID)
.Tk.subwin(parent)
.TkRoot
.TkUp

```

tkdestroy(win)
is.tkwin(x)

tclvalue(x)
tclvalue(x) <- value

tclVar(init = "")
## S3 method for class 'tclVar'
as.character(x, ...)
## S3 method for class 'tclVar'
tclvalue(x)
## S3 replacement method for class 'tclVar'
tclvalue(x) <- value

tclArray()
## S3 method for class 'tclArray'
x[[...]]
## S3 replacement method for class 'tclArray'
x[[...]] <- value
## S3 method for class 'tclArray'
x$i
## S3 replacement method for class 'tclArray'
x$i <- value

## S3 method for class 'tclArray'
names(x)
## S3 method for class 'tclArray'
length(x)

tclObj(x)
tclObj(x) <- value
## S3 method for class 'tclVar'
tclObj(x)
## S3 replacement method for class 'tclVar'
tclObj(x) <- value

as.tclObj(x, drop = FALSE)
is.tclObj(x)

## S3 method for class 'tclObj'
as.character(x, ...)
## S3 method for class 'tclObj'
as.integer(x, ...)
## S3 method for class 'tclObj'
as.double(x, ...)
## S3 method for class 'tclObj'
as.logical(x, ...)
## S3 method for class 'tclObj'
as.raw(x, ...)
## S3 method for class 'tclObj'
tclvalue(x)

```

```
## Default S3 method:
tclvalue(x)
## Default S3 replacement method:
tclvalue(x) <- value
```

```
addTclPath(path = ".")
tclRequire(package, warn = TRUE)
tclVersion()
```

```
objv
win
x
i
drop
value          tclvaluetclobjtclobj
ID
parent
path
package
warn
...
init
```

```
.Tcltclobj.Tcl.objvtclobj
.Tcl.argstag = value-option valuetag = NULL.Tcl.callback-
.Tcl.args.objv.Tcl.argstclobj.Tcl.objv.Tcl.objv
.Tcl.callbackbreak.Tcl.callback"call"R_call 0x408b94d4R_call_lang 0x8a95904
0x819bfd0.Tcl.args
tkwinIDenvIDenvparentnum.subwin.TkRootparent.TkRoot
.Tk.IDID.Tk.newwin.Tk.subwin
.TkUpFALSE
tkdestroy
is.tkwin
tclVarinittcclVaras.charactertclvaluetcvaluetcclVartclVar
tclArraytcclArraytcclVar[[$ias.character(i)lengthnames
tclObjtcclvalue"character""double""integer""logical""raw"as.characteras.tclObj
droptclvalueas.charactertclreadtkgetOpenFileas.raw
tclVartclobj(x)as.tclObjtclobj<Tcl>as.charactertclVar
tclRequireaddTclPath"tclobj"FALSEwarn

    strsplit(tclvalue('auto_path'), " ")[[1]]
```

```
tclVersion"8.6.16""9.0.1"info patchlevel
      as.character(tcl('info', 'tclversion'))
tcl_version9.0b1
```

[TkWidgetsTkCommandsTkWidgetcmds](#)
[capabilities\("tcltk"\)](#)

```
tclVersion()

.Tcl("format \"%s\n\" \"Hello, World!\")

f <- function() cat("HI!\n")
.Tcl.callback(f)
.Tcl.args(text = "Push!", command = f) # NB: Different address

xyzy <- tclVar(7913)
tclvalue(xyzy)
tclvalue(xyzy) <- "foo"
as.character(xyzy)
tcl("set", as.character(xyzy))

## Not run:
## These cannot be run by example() but should be OK when pasted
## into an interactive R session with the tcltk package loaded
top <- tktoplevel() # a Tk widget, see Tk-widgets
ls(envir = top$env, all.names = TRUE)

## End(Not run)

ls(envir = .TkRoot$env, all.names = TRUE) # .Tcl.args put a callback ref in here
```

`tclServiceMode`

`tclServiceMode(on = NULL)`

`on`

```
on == NULL
```

```
## see demo(tkcanvas) for an example
oldmode <- tclServiceMode(FALSE)
# Do some work to create a nice picture.
# Nothing will be displayed until...
tclServiceMode(oldmode)
## another idea is to use tkwm.withdraw() ... tkwm.deiconify()
```

TkCommands

```
tcl(...)
tktitle(x)

tktitle(x) <- value

tkbell(...)
tkbind(...)
tkbindtags(...)
tkfocus(...)
tklower(...)
tkraise(...)

tkclipboard.append(...)
tkclipboard.clear(...)

tkevent.add(...)
tkevent.delete(...)
tkevent.generate(...)
tkevent.info(...)

tkfont.actual(...)
tkfont.configure(...)
tkfont.create(...)
tkfont.delete(...)
tkfont.families(...)
tkfont.measure(...)
tkfont.metrics(...)
tkfont.names(...)
```

```
tkgrab(...)
tkgrab.current(...)
tkgrab.release(...)
tkgrab.set(...)
tkgrab.status(...)
```

```
tkimage.create(...)
tkimage.delete(...)
tkimage.height(...)
tkimage.inuse(...)
tkimage.names(...)
tkimage.type(...)
tkimage.types(...)
tkimage.width(...)
```

```
## NB: some widgets also have a selection.clear command,
## hence the "X".
```

```
tkXselection.clear(...)
tkXselection.get(...)
tkXselection.handle(...)
tkXselection.own(...)
```

```
tkwait.variable(...)
tkwait.visibility(...)
tkwait.window(...)
```

```
## wininfo actually has a large number of subcommands,
## but it's rarely used,
## so use tkwininfo("atom", ...) etc. instead.
```

```
tkwininfo(...)
```

```
# Window manager interface
```

```
tkwm.aspect(...)
tkwm.client(...)
tkwm.colormapwindows(...)
tkwm.command(...)
tkwm.deiconify(...)
tkwm.focusmodel(...)
tkwm.frame(...)
tkwm.geometry(...)
tkwm.grid(...)
tkwm.group(...)
tkwm.iconbitmap(...)
tkwm.iconify(...)
tkwm.iconmask(...)
tkwm.iconname(...)
tkwm.iconposition(...)
tkwm.iconwindow(...)
tkwm.maxsize(...)
```

```
tkwm.minsize(...)
tkwm.overrideredirect(...)
tkwm.positionfrom(...)
tkwm.protocol(...)
tkwm.resizable(...)
tkwm.sizefrom(...)
tkwm.state(...)
tkwm.title(...)
tkwm.transient(...)
tkwm.withdraw(...)
```

Geometry managers

```
tkgrid(...)
tkgrid.bbox(...)
tkgrid.columnconfigure(...)
tkgrid.configure(...)
tkgrid.forget(...)
tkgrid.info(...)
tkgrid.location(...)
tkgrid.propagate(...)
tkgrid.rowconfigure(...)
tkgrid.remove(...)
tkgrid.size(...)
tkgrid.slaves(...)
```

```
tkpack(...)
tkpack.configure(...)
tkpack.forget(...)
tkpack.info(...)
tkpack.propagate(...)
tkpack.slaves(...)
```

```
tkplace(...)
tkplace.configure(...)
tkplace.forget(...)
tkplace.info(...)
tkplace.slaves(...)
```

```
## Standard dialogs
tkgetOpenFile(...)
tkgetSaveFile(...)
tkchooseDirectory(...)
tkmessageBox(...)
tkdialog(...)
tkpopup(...)
```

```
## File handling functions
tclfile.tail(...)
tclfile.dir(...)
```



```
tclopen(...)
tclclose(...)
tclputs(...)
tclread(...)
```

```
x
value          tktitle
...            .Tcl.args
```

```
tcl.Tcl.args.objv.Tcl.objvtcl
tktitlewm title
pack configuretkpack.configuretclfile.dir
```

[TclInterfaceTkWidgetsTkWidgetcmds](#)

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded

tt <- tktoplevel()
tkpack(l1 <- tklabel(tt, text = "Heave"), l2 <- tklabel(tt, text = "Ho"))
tkpack.configure(l1, side = "left")

## Try stretching the window and then

tkdestroy(tt)

## End(Not run)
```

tkpager

file.show

tkpager(file, header, title, delete.file)

```
file
header
title          header
delete.file
```

"\b_"

[file.show](#)

tkProgressBar

```
tkProgressBar(title = "R progress bar", label = "",
              min = 0, max = 1, initial = 0, width = 300)
```

```
getTkProgressBar(pb)
setTkProgressBar(pb, value, title = NULL, label = NULL)
## S3 method for class 'tkProgressBar'
close(con, ...)
```

```
titlelabel
minmax
initialvalue
width
pbcon      "tkProgressBar"
...
```

```
tkProgressBar
setTkProgressBarNULLNAvalue
close
ttk::progressbarprogressbar
```

```
tkProgressBar"tkProgressBar"
getTkProgressBarsetTkProgressBarsetTkProgressBar
```

[txtProgressBar](#)

```
pb <- tkProgressBar("test progress bar", "Some information in %",
                    0, 100, 50)
Sys.sleep(0.5)
u <- c(0, sort(runif(20, 0, 100)), 100)
for(i in u) {
  Sys.sleep(0.1)
  info <- sprintf("%d%% done", round(i))
  setTkProgressBar(pb, i, sprintf("test (%s)", info), info)
}
Sys.sleep(5)
close(pb)
```

tkStartGUI

tkStartGUI()

help()

tkStartGUI().GUIenvTermMenuToolbars(envir = .GUIenv)

TkWidgetcmds

tkactivate(widget, ...)
tkadd(widget, ...)
tkaddtag(widget, ...)
tkbbox(widget, ...)
tkcanvasx(widget, ...)
tkcanvasy(widget, ...)
tkcget(widget, ...)
tkcompare(widget, ...)
tkconfigure(widget, ...)
tkcoords(widget, ...)
tkcreate(widget, ...)
tkcurselection(widget, ...)
tkdchars(widget, ...)
tkdebug(widget, ...)
tkdelete(widget, ...)
tkdelta(widget, ...)
tkdeselect(widget, ...)
tkdlineinfo(widget, ...)
tkdtag(widget, ...)
tkdump(widget, ...)
tkentrycget(widget, ...)
tkentryconfigure(widget, ...)
tkfind(widget, ...)
tkflash(widget, ...)

```
tkfraction(widget, ...)
tkget(widget, ...)
tkgettags(widget, ...)
tkicursor(widget, ...)
tkidentify(widget, ...)
tkindex(widget, ...)
tkinsert(widget, ...)
tkinvoke(widget, ...)
tkitembind(widget, ...)
tkitemcget(widget, ...)
tkitemconfigure(widget, ...)
tkitemfocus(widget, ...)
tkitemlower(widget, ...)
tkitemraise(widget, ...)
tkitemscale(widget, ...)
tkmark.gravity(widget, ...)
tkmark.names(widget, ...)
tkmark.next(widget, ...)
tkmark.previous(widget, ...)
tkmark.set(widget, ...)
tkmark.unset(widget, ...)
tkmove(widget, ...)
tknearest(widget, ...)
tkpost(widget, ...)
tkpostcascade(widget, ...)
tkpostscript(widget, ...)
tkscan.mark(widget, ...)
tkscan.dragto(widget, ...)
tksearch(widget, ...)
tksee(widget, ...)
tkselect(widget, ...)
tkselection.adjust(widget, ...)
tkselection.anchor(widget, ...)
tkselection.clear(widget, ...)
tkselection.from(widget, ...)
tkselection.includes(widget, ...)
tkselection.present(widget, ...)
tkselection.range(widget, ...)
tkselection.set(widget, ...)
tkselection.to(widget, ...)
tkset(widget, ...)
tksize(widget, ...)
tktoggle(widget, ...)
tktag.add(widget, ...)
tktag.bind(widget, ...)
tktag.cget(widget, ...)
tktag.configure(widget, ...)
tktag.delete(widget, ...)
tktag.lower(widget, ...)
tktag.names(widget, ...)
tktag.nextrange(widget, ...)
tktag.prevrange(widget, ...)
```

```

tktag.raise(widget, ...)
tktag.ranges(widget, ...)
tktag.remove(widget, ...)
tktype(widget, ...)
tkunpost(widget, ...)
tkwindow.cget(widget, ...)
tkwindow.configure(widget, ...)
tkwindow.create(widget, ...)
tkwindow.names(widget, ...)
tkxview(widget, ...)
tkxview.moveto(widget, ...)
tkxview.scroll(widget, ...)
tkyposition(widget, ...)
tkyview(widget, ...)
tkyview.moveto(widget, ...)
tkyview.scroll(widget, ...)

```

widget

... .Tcl.args

.a.b selection clear tkselection.clear

TclInterfaceTkWidgetsTkCommands

```

## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded

tt <- tktoplevel()
tkpack(txt.w <- tktext(tt))
tkinsert(txt.w, "0.0", "plot(1:10)")

# callback function
eval.txt <- function() eval(str2lang(tclvalue(tkget(txt.w, "0.0", "end"))))
tkpack(but.w <- tkbutton(tt, text = "Submit", command = eval.txt))

## Try pressing the button, edit the text and when finished:

tkdestroy(tt)

## End(Not run)

```

TkWidgets

tkwidget(parent, type, ...)

tkbutton(parent, ...)
tkcanvas(parent, ...)
tkcheckboxbutton(parent, ...)
tkentry(parent, ...)
ttkentry(parent, ...)
tkframe(parent, ...)
tklabel(parent, ...)
tklistbox(parent, ...)
tkmenu(parent, ...)
tkmenubutton(parent, ...)
tkmessage(parent, ...)
tkradiobutton(parent, ...)
tkscale(parent, ...)
tkscrollbar(parent, ...)
tktext(parent, ...)
tktoplevel(parent = .TkRoot, ...)

ttkbutton(parent, ...)
ttkcheckboxbutton(parent, ...)
ttkcombobox(parent, ...)
ttkframe(parent, ...)
ttklabel(parent, ...)
ttklabelframe(parent, ...)
ttkmenubutton(parent, ...)
ttknotebook(parent, ...)
ttkpanedwindow(parent, ...)
ttkprogressbar(parent, ...)
ttkradiobutton(parent, ...)
ttkscale(parent, ...)
ttkscrollbar(parent, ...)
ttkseparator(parent, ...)
ttksizegrip(parent, ...)
ttkspinbox(parent, ...)
ttktreeview(parent, ...)

parent

type

... [.Tcl.args](#)

```
tkwidgettkwidgettype  
ttkhttps://tkdocs.com/
```

TclInterfaceTkCommandsTkWidgetcmds

```
## Not run:  
## These cannot be run by examples() but should be OK when pasted  
## into an interactive R session with the tcltk package loaded  
  
tt <- tktoplevel()  
label.widget <- tklabel(tt, text = "Hello, World!")  
button.widget <- tkbutton(tt, text = "Push",  
                           command = function()cat("OW!\n"))  
tkpack(label.widget, button.widget) # geometry manager  
                                   # see Tk-commands  
  
## Push the button and then...  
  
tkdestroy(tt)  
  
## test for themed widgets  
if(as.character(tcl("info", "tclversion")) >= "8.5") {  
  # make use of themed widgets  
  # list themes  
  themes <- as.character(tcl("ttk::style", "theme", "names"))  
  themes  
  # select a theme -- for pre-XP windows  
  # tcl("ttk::style", "theme", "use", "winnative")  
  tcl("ttk::style", "theme", "use", themes[1])  
} else {  
  # use Tk 8.0 widgets  
}  
  
## End(Not run)
```

```
tk_choose.dir
```

```
tk_choose.dir(default = "", caption = "Select directory")
```

```
default  
caption
```

NA

[tk_choose.files](#)

```
if (interactive()) tk_choose.dir(getwd(), "Choose a suitable folder")
```

tk_choose.files

```
tk_choose.files(default = "", caption = "Select files",  
                multi = TRUE, filters = NULL, index = 1)
```

default
caption
multi
filters
index

[file.choose](#) [tk_choose.files](#) [file.choose](#)
filters "*" ""

[file.choose](#) [tk_choose.dir](#)

```
Filters <- matrix(c("R code", ".R", "R code", ".s",  
                  "Text", ".txt", "All files", "*"),  
                 4, 2, byrow = TRUE)  
Filters  
if(interactive()) tk_choose.files(filter = Filters)
```

tk_messageBox

```
tk_messageBox(type = c("ok", "okcancel", "yesno", "yesnocancel",  
                      "retrycancel", "abortretryignore"),  
             message, caption = "", default = "", ...)
```

```
type  
message  
caption  
default  
...          icon = "warning"
```

[tkmessageBox](#)

tk_select.list

```
tk_select.list(choices, preselect = NULL, multiple = FALSE,  
              title = NULL)
```

```
choices  
preselect      NULL  
multiple  
title          NULL
```

```
select.listOKCancel  
OK
```

```
multipleCancel""multipleCancel
```

```
select.listmenugraphics = TRUE
```

tools

tools-package

```
library(help = "tools")
```

```
<R-core@r-project.org>
```

.print.via.format

```
.print.via.formatprint()
  print.<myS3class> <- .print.via.format

.print.via.format(x, ...)

x
...                format

xinvisible()print

printprint.default"numeric""character"arraylist
```

```
## The function is simply defined as
function (x, ...) {
  writelines(format(x, ...))
  invisible(x)
}

## is used for simple print methods in R, and as prototype for new methods.
```

add_datalist	datalist
--------------	----------

`data()` data/data/datalist

`add_datalist(pkgpath, force = FALSE, small.size = 1024^2)`

pkgpath	
force	data/datalist
small.size	data/datalistsmall.size

R CMD build data
data/datalist

`data`

assertCondition

```
assertError(expr, classes = "error", verbose = FALSE)
assertWarning(expr, classes = "warning", verbose = FALSE)
assertCondition(expr, ..., .exprString = , verbose = FALSE)
```

expr	<code>tryCatch</code> (expr, ..)
classes...	<code>character</code> "error""warning"
.exprString	exprassertCondition()
verbose	TRUE

```

assertCondition()expr
assertError()assertCondition()
assertWarning()assertCondition(expr, "warning")

```

conditionMessage

stopwarningsignalConditiontryCatch

```

assertError(sqrt("abc"))
assertWarning(matrix(1:8, 4,3))

assertCondition( ""-1 ) # ok, any condition would satisfy this

try( assertCondition(sqrt(2), "warning") )
## .. Failed to get warning in evaluating sqrt(2)
    assertCondition(sqrt("abc"), "error") # ok
try( assertCondition(sqrt("abc"), "warning") )# -> error: had no warning
    assertCondition(sqrt("abc"), "error")
    ## identical to assertError() call above

assertCondition(matrix(1:5, 2,3), "warning")
try( assertCondition(matrix(1:8, 4,3), "error") )
## .. Failed to get expected error ....

## either warning or worse:
assertCondition(matrix(1:8, 4,3), "error","warning") # OK
assertCondition(matrix(1:8, 4, 3), "warning") # OK

## when both are signalled:
ff <- function() { warning("my warning"); stop("my error") }
    assertCondition(ff(), "warning")
## but assertWarning does not allow an error to follow
try(assertWarning(ff()))
    assertCondition(ff(), "error") # ok
assertCondition(ff(), "error", "warning") # ok (quietly, catching warning)

## assert that assertC..() does not assert [and use *one* argument only]
assertCondition( assertCondition(sqrt( 2 ), "warning") )
assertCondition( assertCondition(sqrt("abc"), "warning"), "error")
assertCondition( assertCondition(matrix(1:8, 4,3), "error"),
                "error")

```

basetools

```
base_aliases_db(verbose = FALSE, Ncpus = getOption("Ncpus", 1L))
base_rdxrefs_db(verbose = FALSE, Ncpus = getOption("Ncpus", 1L))
```

```
verbose
Ncpus
```

```
base_aliases_db()
base_rdxrefs_db() "Target" "Anchor" "Source"
```

```
CRAN\_aliases\_db\(\) CRAN\_rdxrefs\_db\(\)
```

bibstyle

```
bibentryRd
```

```
bibstyle(style, envir, ..., .init = FALSE, .default = TRUE)
getBibstyle(all = FALSE)
```

```
style
envir
...
.init      "JSS"
.default
all
```

```
bibentry
```

```
.init = TRUE "JSS" jss.bstenvir
getBibstyle() bibstyle(style, .default = FALSE) style... "JSS" style NULL
bibentry formatArticle formatBook formatInbook formatIncollection
formatInProceedings formatManual formatMastersthesis formatMisc formatPhdthesis
formatProceedings formatTechreport formatUnpublished unclassbibentry
sortKeys bibentry citeutils::cite
format "bibentry" ".index" "JSS" fmtPrefix()
```

bibstyle
getBibstyle

[bibentry](#)

```
refs <-  
c(bibentry(bibtype = "manual",  
  title = "R: A Language and Environment for Statistical Computing",  
  author = person("R Core Team"),  
  organization = "R Foundation for Statistical Computing",  
  address = "Vienna, Austria",  
  year = 2013,  
  url = "https://www.R-project.org"),  
  bibentry(bibtype = "article",  
    author = c(person(c("George", "E.", "P."), "Box"),  
      person(c("David", "R."), "Cox")),  
    year = 1964,  
    title = "An Analysis of Transformations",  
    journal = "Journal of the Royal Statistical Society, Series B",  
    volume = 26, number = 2, pages = "211--243",  
    doi = "10.1111/j.2517-6161.1964.tb00553.x"))  
  
bibstyle("unsorted", sortKeys = function(refs) seq_along(refs),  
  fmtPrefix = function(paper) paste0("[", paper$.index, "]"),  
  .init = TRUE)  
print(refs, .bibstyle = "unsorted")
```

buildVignette

[Sweavetexi2pdfStangle](#)

R CMD Sweave

```
buildVignette(file, dir = ".", weave = TRUE, latex = TRUE, tangle = TRUE,  
  quiet = TRUE, clean = TRUE, keep = character(),  
  engine = NULL, buildPkg = NULL, encoding, ...)
```

file
dir
weave
latex .tex

```
tangle
quiet
clean
keep
engine      NULL\VignetteEngine{}
buildPkg     NULL
encoding     buildVignettesDESCRIPTION
...

utils::SweavecleanTRUEkeepcleanNAweave.tex.pdf
buildPkg\VignetteEngine{utils::Sweave}Sweaveengine
```

buildVignettes

buildVignettes

Sweavetexi2pdf

```
buildVignettes(package, dir, lib.loc = NULL, quiet = TRUE,
               clean = TRUE, tangle = FALSE, skip = NULL,
               ser_elibs = NULL)

pkgVignettes(package, dir, subdirs = NULL, lib.loc = NULL,
             output = FALSE, source = FALSE, check = FALSE)

package      doc
dir           vignettes
lib.loc       NULLNULLpackage
quiet         texi2pdf
clean
tangle
skip          namespkgVignettesTRUE\VignetteDependsvignetteInfo
ser_elibs     R CMD check
subdirs       dir"doc"package"vignettes"
output        outputs
source        sources
check         TRUE
```

```
buildVignettesR CMD buildR CMD check
.Rbuildignoredir
```

```
buildVignettestangle = TRUE
pkgVignettes"pkgVignettes"NULL
```

```
gVigns <- pkgVignettes("grid")
str(gVigns)
```

charsets

```
charset_to_Unicode
Adobe_glyphs"adobe""unicode"
```

```
charset_to_Unicode
Adobe_glyphs
```

```
charset_to_Unicode("noquote", "hexmode")libiconv
Adobe_glyphs/share/encodings/Adobe_glyphlist
```

```
## find Adobe names for ISOLatin2 chars.
latin2 <- charset_to_Unicode[, "ISOLatin2"]
aUnicode <- as.hexmode(paste0("0x", Adobe_glyphs$unicode))
keep <- aUnicode %in% latin2
aUnicode <- aUnicode[keep]
aAdobe <- Adobe_glyphs[keep, 1]
## first match
aLatin2 <- aAdobe[match(latin2, aUnicode)]
## all matches
bLatin2 <- lapply(1:256, function(x) aAdobe[aUnicode == latin2[x]])
format(bLatin2, justify = "none")
```

checkFF

.C.Fortran"NativeSymbolInfo"PACKAGE

```
checkFF(package, dir, file, lib.loc = NULL,  
        registration = FALSE, check_DUP = FALSE,  
        verbose = getOption("verbose"))
```

package

dir	Rpackage
file	packagedir
lib.loc	NULLNULLpackage
registration	TRUE
check_DUP	TRUE.C.FortranDUP = FALSE
verbose	TRUE

name

useDynLibuseDynLib

PACKAGEDESCRIPTION

"checkFF"

formatprint

.C.FortranForeign

```
# order is pretty much random  
checkFF(package = "stats", verbose = TRUE)
```

checkMD5sums

checkMD5sumsMD5

checkMD5sums(package, dir)

package

dir

MD5md5sum -c MD5md5sum<https://cran.r-project.org/bin/windows/Rtools/>

dirpackage

tools:::.installMD5sumsMD5

checkMD5sumsNAMD5

[md5sum](#)

checkPoFiles

checkPoFilecheckPoFiles

checkPoFile(f, strictPlural = FALSE)

checkPoFiles(language, dir = ".")

f

strictPlural

language dir""dir

dir

[.posprintf](#)

cupnsprintf()

strictPluralTRUEFALSEcheckPoFilescheckPoFilestrictPluralTRUEFALSE

.po

"\uFF05"

```
"check_po_files"print
```

```
.po  
https://www.gnu.org/software/gettext/manual/gettext.html
```

```
update_pkg_po()checkPoFile()xgettextsprintf
```

```
## Not run:  
checkPoFiles("de", "/path/to/R/src/directory")  
  
## End(Not run)
```

```
checkRd
```

```
parse_Rd
```

```
checkRd(Rd, defines = .Platform$OS.type, stages = "render",  
        unknownOK = TRUE, listOK = TRUE, ..., def_enc = FALSE)
```

Rd	Rd
defines	#ifdef
stages	"build""install""render"\Sexpr
unknownOK	FALSE
listOK	FALSE
...	parse_Rd Rdencoding
def_enc	

```
checkRd  
parse\_Rdparse_Rdprepare_Rd#ifdef\Sexpr  
prepare_Rd  
prepare_Rd  
  
\Sexpr  
\Sexpr
```

checkRd

```
\tabular
\tabular
```

```
\method
\method\usage
\dontrun\examples
```

```
\name
\item
```

```
\name
```

```
\enc
```

```
\ldots
\title
```

```
\method\S3method\S4method\dontrun\donttest\dontdiff\dontshow\title\section
\subsectioncode{text}codetext
```

```
prepare_RdcheckRdcheckRd
```

```
"checkRd"printminlevel-1R CMD check
```

```
!unknownOK!listOK\if\ifelse\Sexprprepare_Rd\Rdversion\encoding\docType\name
\name\title\Sexprstages
```

[parse_RdRd2HTML](#)

```
## parsed Rd from the installed version of _this_ help file
rd <- Rd_db("tools")["checkRd.Rd"]
rd
stopifnot(length(checkRd(rd)) == 0) # there should be no issues

## make up \tabular issues
bad <- r"(\name{bad}\title{bad}\description{\tabular{p}{1 \tab 2}})"
(res <- checkRd(parse_Rd(textConnection(bad))))
stopifnot(length(res) > 0)
```

checkRdaFiles

```
save
.rda.RData
```

```
checkRdaFiles(paths)
resaveRdaFiles(paths, compress = c("auto", "gzip", "bzip2", "xz"),
               compression_level, version = NULL)
```

```
paths          save.rda.RData
compresscompression_level
               savecompress
version        save
```

```
compress = "auto"compression_level"gzip""bzip2""gzip""xz""gzip"
version = NULL
```

```
checkRdaFilespaths
```

```
size          NA
ASCII         NA
compress      "gzip""bzip2""xz""none""unknown"
version       save()NA
```

```
## Not run:
## from a package top-level source directory
paths <- sort(Sys.glob(c("data/*.rda", "data/*.RData")))
(res <- checkRdaFiles(paths))
## pick out some that may need attention
bad <- is.na(res$ASCII) | res$ASCII | (res$size > 1e4 & res$compress == "none")
res[bad, ]

## End(Not run)
```

checkTnF

TFTTRUEFALSETRUEFALSE

checkTnF(package, dir, file, lib.loc = NULL)

package	
dir	Rmanpackage
file	packagedir
lib.loc	NULLNULLpackage

"checkTnF"TF
print

checkVignettes

[SweaveStanglesource](#)

```
checkVignettes(package, dir, lib.loc = NULL,  
               tangle = TRUE, weave = TRUE, latex = FALSE,  
               workdir = c("tmp", "src", "cur"),  
               keepfiles = FALSE)
```

package	doc
dir	vignettes
lib.loc	NULLNULLpackage
tangle	source
weave	
latex	weavelatexTRUEMakefile.tex texi2pdf
workdir	"tmp""src""cur"
keepfiles	workdir != "tmp"

[pkgVignettesvignetteEngine](#)
[tangleStanglesource](#)
weave[SweavelatexMakefiletexi2pdf](#).tex

"checkVignettes"print

check_packages_in_dir

```
check_packages_in_dir(dir,
                      pfiles = Sys.glob("*.tar.gz"),
                      check_args = character(),
                      check_args_db = list(),
                      reverse = NULL,
                      check_env = character(),
                      xvfb = FALSE,
                      Ncpus = getOption("Ncpus", 1L),
                      clean = TRUE,
                      install_args = list(),
                      parallel_args = list(),
                      ...)

summarize_check_packages_in_dir_results(dir, all = TRUE,
                                       full = FALSE, ...)
summarize_check_packages_in_dir_timings(dir, all = FALSE,
                                       full = FALSE)
summarize_check_packages_in_dir_depends(dir, all = FALSE,
                                       which = c("Depends",
                                              "Imports",
                                              "LinkingTo"))
```

```
check_packages_in_dir_changes(dir, old,
                              outputs = FALSE, sources = FALSE, ...)
check_packages_in_dir_details(dir, logs = NULL, drop_ok = TRUE, ...)
```

dir	.tar.gz
pfiles	*.tar.gzdir
check_args	R CMD check
check_args_db	R CMD check
reverse	"repos""which""recursive"getOption("repos")c("Depends", "Imports", "LinkingTo")"most""all" package_dependencies FALSE NULL
check_env	
xvfb	Xvfb
Ncpus	
clean	
install_args	install.packages
parallel_args	parLapplymclapply

```

...          readLines(check_packages_in_dir
all
full
which        package_dependencies
old          check_packages_in_dir
outputs
sources      FALSE
logs         00check.logdir
drop_ok      OKNONESKIPPED

check_packages_in_dir.tar.gzdirreverse
"which"reverse"recursive"reverse
.tar.gzdircleanLibrarydir.tar.gzOutputsdir*.Rcheckrdepends_
summarize_check_packages_in_dir_resultssummarize_check_packages_in_dir_timings
NcpusmclapplyparLapply
check_packages_in_dir"check_packages_in_dir"printsummary
check_packages_in_dir_changesdioldoutputssources"=="!="<"<=">">="
check_packages_in_dir_detailsCheckStatusOutput
_R_CHECK_ELAPSED_TIMEOUT_checkcheck

```

```

## Not run:
## Check packages in dir without reverse dependencies:
check_packages_in_dir(dir)
## Check packages in dir and their reverse dependencies using the
## defaults (all repositories in getOption("repos"), all "strong"
## reverse dependencies, no recursive reverse dependencies):
check_packages_in_dir(dir, reverse = list())
## Check packages in dir with their reverse dependencies from CRAN,
## using all strong reverse dependencies and reverse suggests:
check_packages_in_dir(dir,
                      reverse = list(repos = getOption("repos")["CRAN"],
                                     which = "most"))
## Check packages in dir with their reverse dependencies from CRAN,
## using '--as-cran' for the former but not the latter:
check_packages_in_dir(dir,
                      check_args = c("--as-cran", ""),
                      reverse = list(repos = getOption("repos")["CRAN"]))

## End(Not run)

```

codoc

codoccodocClassescodocData

```
codoc(package, dir, lib.loc = NULL,  
      use.values = NULL, verbose = getOption("verbose"))  
codocClasses(package, lib.loc = NULL)  
codocData(package, lib.loc = NULL)
```

package

dir	manRpackage
lib.loc	NULLNULLpackage
use.values	FALSETRUE
verbose	TRUE

codoc

use.values

[prompt](#)codocDatacodocClasses[promptClass](#)

-defunct.Rd

codoc"codoc"codedocs

codocClassescodocData"codocClasses""codocData""codoc"

print

use.valuesFALSENULL

[undocQC](#)

compactPDF

R CMD build --compact-vignettes

```
compactPDF(paths,  
            qpdf = Sys.which(Sys.getenv("R_QPDF", "qpdf")),  
            gs_cmd = Sys.getenv("R_GSCMD", ""),  
            gs_quality = Sys.getenv("GS_QUALITY", "none"),  
            gs_extras = character(),  
            verbose = FALSE)
```

```
## S3 method for class 'compactPDF'  
format(x, ratio = 0.9, diff = 1e4, ...)
```

paths	.pdf
qpdf	qpdfqpdf
gs_cmd	gswin32c.exe gswin64c.exe"
gs_quality	"none" "printer" "ebook" "screen"
gs_extras	
verbose	logical
x	"compactPDF"
ratiodiff	ratiodiff
...	

```
qpdf https://qpdf.sourceforge.io/gs_cmdgs_quality != "none"qpdfgs_quality !=  
"none"gs_cmd"
```

```
qpdfgs_cmd  
qpdf"ebook"
```

```
pdflatex  
qpdfgs_quality != "none"
```

```
c("compactPDF", "data.frame")  
formatprint...ratiodiff
```

[resaveRdaFiles](#)

CRANtools

CRAN_package_db()

CRAN_check_results(flavors = NULL)

CRAN_check_details(flavors = NULL)

CRAN_check_issues()

summarize_CRAN_check_status(packages,
 results = NULL,
 details = NULL,
 issues = NULL)

CRAN_current_db()

CRAN_aliases_db()

CRAN_rdxrefs_db()

CRAN_archive_db()

CRAN_authors_db()

packages

flavors NULL

results CRAN_check_results()

details CRAN_check_details()

issues CRAN_check_issues()

CRAN_package_db()DESCRIPTIONutils::available.packages()

CRAN_check_results()PackageFlavorStatus

CRAN_check_details()"check_details"printformatCheckStatusOutput*CheckStatus

CRAN_check_issues()<https://www.stats.ox.ac.uk/pub/bdr/memtests/PackageVersion>
kindhref

CRAN_current_db()[file.info\(\)](#)

CRAN_aliases_db()

CRAN_rdxrefs_db()"Target""Anchor""Source"

CRAN_archive_db()[file.info\(\)](#)

CRAN_authors_db()Authors@RDESCRIPTION

aAbB

delimMatch

```
delimMatch(x, delim = c("{", "}"), syntax = "Rd")
```

```
x  
delim  
syntax      "Rd"%\
```

```
x-1"match.length"-1
```

regexpr

```
x <- c("\\value{foo}", "function(bar)")  
delimMatch(x)  
delimMatch(x, c("(", ")"))
```

dependsOnPkgs

```
dependsOnPkgs(pkgs,  
              dependencies = "strong",  
              recursive = TRUE, lib.loc = NULL,  
              installed =  
                utils::installed.packages(lib.loc, fields = "Enhances"))
```

```
pkgs  
dependencies  c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")  
              "all" "most" "Enhances" "strong"  
recursive  
lib.loc      NULL.libPaths  
installed    installed.packages
```

```
pkgs
```

```
package_dependencies()
```

```
## there are few dependencies in a vanilla R installation:  
## lattice may not be installed  
dependsOnPkgs("lattice")
```

```
doitools
```

```
check_package_dois(dir, verbose = FALSE)
```

```
dir  
verbose
```

```
DESCRIPTION<doi:...>DescriptionCITATION  
HEAD
```

```
check_doi_db
```

```
encoded_text_to_latex
```

```
encoded_text_to_latex(x,  
                      encoding = c("latin1", "latin2", "latin9",  
                                   "UTF-8", "utf8"))
```

```
x  
encoding      "latin9"inputenc
```

x?

\-

x

iconv

```
x <- "fran\xE7ais"
encoded_text_to_latex(x, "latin1")
## Not run:
## create a tex file to show the upper half of 8-bit charsets
x <- rawToChar(as.raw(160:255), multiple = TRUE)
(x <- matrix(x, ncol = 16, byrow = TRUE))
xx <- x
xx[] <- encoded_text_to_latex(x, "latin1") # or latin2 or latin9
xx <- apply(xx, 1, paste, collapse = "&")
con <- file("test-encoding.tex", "w")
header <- c(
  "\\documentclass{article}",
  "\\usepackage[T1]{fontenc}",
  "\\usepackage{Rd}",
  "\\begin{document}",
  "\\HeaderA{test}{test}",
  "\\begin{Details}\\relax",
  "\\Tabular{cccccccccccccc}{")
trailer <- c(")", "\\end{Details}", "\\end{document}")
writelines(header, con)
writelines(paste0(xx, "\\\""), con)
writelines(trailer, con)
close(con)
## and some UTF_8 chars
x <- intToUtf8(as.integer(
  c(160:383, 0x0192, 0x02C6, 0x02C7, 0x02CA, 0x02D8,
    0x02D9, 0x02DD, 0x200C, 0x2018, 0x2019, 0x201C,
    0x201D, 0x2020, 0x2022, 0x2026, 0x20AC)),
  multiple = TRUE)
x <- matrix(x, ncol = 16, byrow = TRUE)
xx <- x
xx[] <- encoded_text_to_latex(x, "UTF-8")
xx <- apply(xx, 1, paste, collapse = "&")
con <- file("test-utf8.tex", "w")
writelines(header, con)
writelines(paste(xx, "\\\""), sep = ""), con)
writelines(trailer, con)
close(con)

## End(Not run)
```

fileutils

```
file_ext(x)
file_path_as_absolute(x)
file_path_sans_ext(x, compression = FALSE)

list_files_with_exts(dir, exts, all.files = FALSE,
                     full.names = TRUE)
list_files_with_type(dir, type, all.files = FALSE,
                     full.names = TRUE, OS_subdirs = .OSType())
```

```
x
compression      .gz.bz2.xz
dir
exts
all.files        FALSETRUE
full.names
type             "code""data""demo""docs""vignette"
OS_subdirs       R_OSTYPE.Platform$OS.type
```

```
file_ext
file_path_as_absolutenormalizePathx
file_path_sans_ext
list_files_with_extsdirexts
list_files_with_typedirOS_subdirs
```

[file.path](#)[file.info](#)[list.files](#)

```
dir <- file.path(R.home(), "library", "stats")
list_files_with_exts(file.path(dir, "demo"), "R")
list_files_with_type(file.path(dir, "demo"), "demo") # the same
file_path_sans_ext(list.files(file.path(R.home("modules"))))
```

`find_gs_cmd`

```
find_gs_cmd(gs_cmd = "")
```

```
gs_cmd
```

```
gsgs_cmdR_GSCMDgs
```

```
gs_cmdR_GSCMDGSCgswin64cgswin32cPATH
```

```
## Not run:
```

```
## Suppose a Solaris system has GhostScript 9.00 on the path and
```

```
## 9.07 in /opt/csw/bin. Then one might set
```

```
Sys.setenv(R_GSCMD = "/opt/csw/bin/gs")
```

```
## End(Not run)
```

`getVignetteInfo`

```
getVignetteInfo(package = NULL, lib.loc = NULL, all = TRUE)
```

```
package      NULL
```

```
lib.loc
```

```
all
```

Package
Dir
Topic
File
Title
R
PDF

PDF

[pkgVignettes](#)

`getVignetteInfo("grid")`

HTMLheader

```
HTMLheader(title = "R", logo = TRUE, up = NULL,  
            top = file.path(Rhome, "doc/html/index.html"),  
            Rhome = "",  
            css = file.path(Rhome, "doc/html/R.css"),  
            headerTitle = paste("R:", title),  
            outputEncoding = "UTF-8")
```

title
logo
up
top
Rhome
css
headerTitle
outputEncoding

`uptopRhome../..`

```
cat(HTMLheader("This is a sample header"), sep="\n")
```

HTMLlinks

```
findHTMLlinks(pkgDir = "", lib.loc = NULL, level = 0:3)
```

```
pkgDir  
lib.loc      .libPaths()  
level
```

```
findHTMLlinks
```

```
  pkgDir
```

```
  lib.loc
```

```
html"../../html/.html"
```

licensetools

```
analyze_license(x)
```

```
x
```

```
LicenseDESCRIPTIONLICENSELICENCE
```

```
analyze_license(https://en.wikipedia.org/wiki/Free\_and\_open-source\_software)
```

is_canonical
is_standardizable

standardization

components
expansions
is_verified

```
## Examples from section 'Licenses' of 'Writing R Extensions':  
analyze_license("GPL-2")  
analyze_license("LGPL (>= 2.0, < 3) | Mozilla Public License")  
analyze_license("GPL-2 | file LICENCE")  
analyze_license("GPL (>= 2) | BSD_3_clause + file LICENSE")  
analyze_license("Artistic-2.0 | AGPL-3 + file LICENSE")
```

loadRdMacros

.Rd.Rd

```
loadRdMacros(file, macros = TRUE)  
loadPkgRdMacros(pkgdir, macros = NULL)
```

file
macros [parse_Rd](#) macrosloadPkgRdMacros
pkgdir

```
macrosloadRdMacrosloadPkgRdMacros  
loadPkgRdMacros"RdMacros"DESCRIPTION.Rdman/macros  
help/macros
```

[emptyenv\(\)](#)

<https://developer.r-project.org/parseRd.pdf>

parse_Rd

```
f <- tempfile()
writeLines(r"(
\newcommand{\Rlogo}{
  \if{html}{\figure{Rlogo.svg}{options: width=100 alt="R logo"}}
  \if{latex}{\figure{Rlogo.pdf}{options: width=0.5in}}
}
)", f)
m <- loadRdMacros(f)
ls(m)
ls(parent.env(m))
ls(parent.env(parent.env(m)))
parse_Rd(textConnection(r"(\Rlogo)"), fragment = TRUE, macros = m)
```

makevars

Makevars

```
makevars_user()
makevars_site()
```

```
srcctat(tools::makevars_user())cat(tools::makevars_site())-f
```

Makevars

```
makevars_user()
makevars_site()
```

`make_translations_pkg`

```
make_translations_pkg(srcdir, outDir = ".", append = "-1")
```

```
srcdir
```

```
outDir
```

```
append          3.0.0-1
```

```
update.packagesR-patched
```

```
Depends3.x.*x
```

`matchConcordance`

```
.RdRd2HTMLRd2latex
```

```
matchConcordance"Rconcordance"as.characteras.RconcordancematchConcordance
```

```
matchConcordance(linenum, concordance)
## S3 method for class 'Rconcordance'
as.character(x, targetfile = "", ...)
as.Rconcordance(x, ...)
followConcordance(concordance, prevConcordance)
```

```
linenum
```

```
concordance      "Rconcordance"
```

```
prevConcordance
```

```
targetfile
```

```
x                as.character"Rconcordance"as.concordanceas.character
```

```
...
```

```
matchConcordance
"Rconcordance"as.character
as.Rconcordanceas.character.Rconcordance
followConcordanceprevConcordancefollowConcordanceconcordance
```

```
"activeConcordance"Rd2HTMLRd2latex
"Rconcordance"
offset
srcLine srcLine
srcFile srcLine
"Rconcordance"as.characteras.concordance"Rconcordance"
```

```
matchConcordancelinenum"srcFile""srcLine"
"Rconcordance"as.characterSweave
as.concordance"Rconcordance"NULL
```

[Rd2HTMLRd2latex](#)

md5sum

```
md5sum(files, bytes)
```

```
files
bytes          bytesfiles
```

```
md5sum
```

filesfilesfilesNA

bytes

glibc

[checkMD5sumsha256sum](#)

```
as.vector(md5sum(dir(R.home(), pattern = "^COPY", full.names = TRUE)))
md5sum(bytes=raw())
md5sum(bytes=charToRaw("abc"))
```

package_dependencies

```
package_dependencies(packages = NULL, db = NULL, which = "strong",
                      recursive = FALSE, reverse = FALSE,
                      verbose = getOption("verbose"))
```

packages

db [available.packages\(\)](#)NULL<https://cran.r-project.org/web/packages/packages.rds>

which c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")
 "all""most""Enhances""strong"

recursive which

reverse FALSE

verbose

packages

NULLcharacter(0)

[dependsOnPkgs](#)


```

myPkgs <- c("MASS", "Matrix", "KernSmooth", "class", "cluster", "codetools")
pdb <- available.packages(repos = findCRANmirror("web"))
system.time(
  dep1 <- package_dependencies(myPkgs, db = pdb) # all arguments at default
) # very fast
utils::str(dep1, vec.len=10)

system.time( ## reverse dependencies, recursively --- takes much longer:
  deps <- package_dependencies(myPkgs, db = pdb, which = "most",
                             recursive = TRUE, reverse = TRUE)
) # seen ~ 10 seconds

lengths(deps) # 2020-05-03: all are 16053, but codetools with 16057

## install.packages(dependencies = TRUE) installs 'most' dependencies
## and the strong recursive dependencies of these: these dependencies
## can be obtained using 'which = "most"' and 'recursive = "strong"'.
## To illustrate on the first packages with non-missing Suggests:
packages <- pdb[head(which(!is.na(pdb[, "Suggests"]))), "Package"]
package_dependencies(packages, db = pdb,
                    which = "most", recursive = "strong")

```

package_native_routine_registration_skeleton

```

package_native_routine_registration_skeleton(dir, con = stdout(),
  align = TRUE, character_only = TRUE, include_declarations = TRUE)

```

```

dir
con
align
character_only .NAME
include_declarations

```

```

src/init.c
R.C.Fortran.Call.External-1
.Call.External.C.Fortranvoid *init.c
character_onlycharacter_only = FALSE.NAME
useDynLibcharacter_only = FALSE
.External-1

.Call(...)

```

con

cproto<https://invisible-island.net/cproto/cproto.html>

```
cproto -I/path/to/R/include -e *.c
```

ctags

```
ctags -x *.c
```

gfortran

```
gfortran -c -fc-prototypes-external file.f
```

bind(C)

R.Call

```
.Cif(FALSE) { ... }
```

```
.Call(if(int) "rle_i" else "rle_d", i, force)
```

```
.Call (cfunction, ...)
```

```
.Call(..., PACKAGE="sparseLTSEigen")
```

character_only" cfunction"

PACKAGE

[package.skeleton](#)

```

## Not run:
## with a completed splines/DESCRIPTION file,
tools::package_native_routine_registration_skeleton('splines',,,FALSE)
## produces
#include <R.h>
#include <Rinternals.h>
#include <stdlib.h> // for NULL
#include <R_ext/Rdynload.h>

/* FIXME:
   Check these declarations against the C/Fortran source code.
*/

/* .Call calls */
extern SEXP spline_basis(SEXP, SEXP, SEXP, SEXP);
extern SEXP spline_value(SEXP, SEXP, SEXP, SEXP, SEXP);

static const R_CallMethodDef CallEntries[] = {
    {"spline_basis", (DL_FUNC) &spline_basis, 4},
    {"spline_value", (DL_FUNC) &spline_value, 5},
    {NULL, NULL, 0}
};

void R_init_splines(DllInfo *dll)
{
    R_registerRoutines(dll, NULL, CallEntries, NULL, NULL);
    R_useDynamicSymbols(dll, FALSE);
}

## End(Not run)

```

parseLatex

parseLatexdeparseLatexlatexToUtf8

```

parseLatex(text, filename = "text",
            verbose = FALSE,
            verbatim = c("verbatim", "verbatim*",
                          "Sinput", "Soutput"),
            verb = "\\Sexpr",
            defcmd = c("\\newcommand", "\\renewcommand",
                       "\\providecommand", "\\def", "\\let"),
            defenv = c("\\newenvironment",
                       "\\renewenvironment"))
deparseLatex(x, dropBraces = FALSE)
latexToUtf8(x)

```

```
text
filename
verbose      TRUE
verbatim
verb
defcmddefenv
x            "LaTeX"
dropBraces   "LaTeX"
```

```
\verbverbatimverb
```

```
parseLatex()"LaTeX""latex_tag"
deparseLatex()
latexToUtf8()"LaTeX"
```

```
latex <- parseLatex("fran\\c{c}ais")
deparseLatex(latexToUtf8(latex))
```

```
parse_Rd
```

```
parse_Rd(file, srcfile = NULL, encoding = "unknown",
          verbose = FALSE, fragment = FALSE, warningCalls = TRUE,
          macros = file.path(R.home("share"), "Rd", "macros", "system.Rd"),
          permissive = FALSE)
## S3 method for class 'Rd'
print(x, deparse = FALSE, ...)
## S3 method for class 'Rd'
as.character(x, deparse = FALSE, ...)
```

```
file
srcfile      NULL"srcfile"
encoding
verbose
fragment
warningCalls
macros
permissive
x
deparse      TRUE
...
```

Rd<https://developer.r-project.org/parseRd.pdf>

```
encoding
\newcommand\renewcommandloadRdMacrosloadPkgRdMacroparse_RdFALSE"macros"
permissivebibentrypermissive = TRUEparse_Rd
```

```
parse_Rd"Rd"as.character()print()
macros = FALSE"macros"fileparse_Rd
```

<https://developer.r-project.org/parseRd.pdf>

Rd2HTMLparse_Rd()

```

pkg2HTML(package, dir = NULL, lib.loc = NULL,
  outputEncoding = "UTF-8",
  stylesheet = file.path(R.home("doc"), "html", "R-nav.css"),
  hooks = list(pkg_href = function(pkg) sprintf("%s.html", pkg)),
  texmath = getOption("help.htmlmath"),
  prism = TRUE,
  out = NULL,
  toc_entry = c("title", "name"),
  ...,
  Rhtml = FALSE,
  mathjax_config = file.path(R.home("doc"), "html", "mathjax-config.js"),
  include_description = TRUE)

```

```

package
dir
lib.loc      NULL find.packageRd_db
outputEncoding Rd2HTML
stylesheet
hooks        pkg_href
texmath      "katex""mathjax""katex"mathjaxr
prism        Rd2HTML
out          NULLhooks$pkg_href.html
toc_entry
...          Rd2HTMLstagesRd_db
Rhtml        TRUErcode
mathjax_config texmath = "mathjax"
include_description
              DESCRIPTION

```

```

pkg2HTML
\Sexprstages

```

```

parse_RdRd_dbRd2HTML

```

```

pkg2HTML("tools", out = tempfile(fileext = ".html")) |> browseURL()

```

pskill

pskill

pskill(pid, signal = SIGTERM)

SIGHUP
SIGINT
SIGQUIT
SIGKILL
SIGTERM
SIGSTOP
SIGTSTP
SIGCHLD
SIGUSR1
SIGUSR2

pid [Sys.getpid](#)

signal

SIGINTSIGTERMKillpskill

Ctrl-CSIGINTCtrl-\SIGQUITCtrl-ZSIGTSTPSIGSTOPSIGCONT

SIG*NA_INTEGER_

SIGINTSIGTERMpkillTerminateProcess

pidTRUEFALSE

[psnice](#)

```
## Not run:  
pskill(c(237, 245), SIGKILL)
```

```
## End(Not run)
```

psnice

```
psnice(pid = Sys.getpid(), value = NA_integer_)
```

```
pid
value      NA
```

```
+19
top/usr/bin/renice/usr/bin/nice/usr/bin/renice -n
```

```
19150-5-10
```

```
NA
```

[pskill](#)

QC

```
checkDocFiles (package, dir, lib.loc = NULL, chkInternal = NULL)
checkDocStyle (package, dir, lib.loc = NULL)
checkReplaceFuns(package, dir, lib.loc = NULL)
checkS3methods (package, dir, lib.loc = NULL)
checkRdContents (package, dir, lib.loc = NULL, chkInternal = NULL)
```

```
langEls
nonS3methods(package)
```

```
package
dir      Rmanpackage
lib.loc  NULLNULLpackage
chkInternal  internalNULL
```



```
checkDocFiles\alias
checkDocStyle\method\methodcheckDocStyle
checkReplaceFunsvalue
checkS3methodsformula...UseMethodMath
checkRdContents()
nonS3methods(package)characterpackagepackage = NULL
langEltsprint().Primitive("")
```

```
print
```

R

```
R(fun, args = list(), opts = "--no-save --no-restore",
  env = character(), arch = "", drop = TRUE, timeout = 0)
```

```
fun
args
opts
env
arch
drop          drop = FALSE
timeout
```

```
system2()optsenvarchtimeout
```

```
drop = TRUE
"inferiorCallError"value
```

```
## Compute cos(0) in an inferior R process.
## By default, only return the value of the function call.
R(cos, list(0))
## If 'drop = FALSE', we also get status, stdout and stderr.

R(cos, list(0), drop = FALSE)

## A call giving an error:
(e <- tryCatch(R(stop, list("FOOBAR")), error = identity))
## The inferior R process ran successfully:
e$status
## The function call gave an error:
e$value
```

Rcmd	R CMD
R CMD	
Rcmd(args, ...)	
args	R CMD
...	system2
R CMD	system2
system2	
Rd2HTML	

```
parse\_Rd\(\)Rd

Rd2HTML(Rd, out = "", package = "", defines = .Platform$OS.type,
  Links = NULL, Links2 = NULL,
  stages = "render", outputEncoding = "UTF-8",
  dynamic = FALSE, no_links = FALSE, fragment = FALSE,
  stylesheet = if (dynamic) "/doc/html/R.css" else "R.css",
  texmath = getOption("help.htmlmath"),
  concordance = FALSE,
  standalone = TRUE,
```

```

hooks = list(),
toc = isTRUE(getOption("help.htmltoc")),
Rhtml = FALSE,
...)

Rd2txt(Rd, out = "", package = "", defines = .Platform$OS.type,
stages = "render", outputEncoding = "",
fragment = FALSE, options, ...)

Rd2latex(Rd, out = "", defines = .Platform$OS.type,
stages = "render", outputEncoding = "UTF-8",
fragment = FALSE, ..., writeEncoding = outputEncoding != "UTF-8",
concordance = FALSE)

Rd2ex(Rd, out = "", defines = .Platform$OS.type,
stages = "render", outputEncoding = "UTF-8",
commentDontrun = TRUE, commentDonttest = FALSE, ...)

```

Rd	Rd
out	out = ""out = stdout()
package	
defines	<code>#ifdef</code>
stages	"build""install""render"\Sexpr
outputEncoding	
dynamic	
no_links	R CMD Rdconv
fragment	
stylesheet	
texmath	<code>\eqn\deqn"katex""mathjax"</code> mathjaxr
concordance	
standalone	FALSE
hooks	<code>pkg_hrefpkg2HTML</code>
toc	standalone = TRUE
Rhtml	TRUErcode
LinksLinks2	NULL findHTMLlinks
options	Rd2txt_options
...	parse_RdRd
writeEncoding	<code>\inputencoding</code>
commentDontrun	<code>\dontrun</code>
commentDonttest	<code>\donttest</code>

```

Rd2HTMLRd2txtRd2latexRd2exexample
parse_Rd
LinkLink2
Rd2latexoutputEncoding = "ASCII"\enc
Rd2txtwidth
Rd2txtsQuote"useFancyQuotes"sQuote
Rd2txtoptionsRd2txt_optionsoptionsRd2txt_options
fragment = TRUERd\Sexpr#ifdef#ifndef

```

```

Rd2latex"latexEncoding"inputencRd2HTMLstandalone = FALSE"info"nametitle
Rd2HTMLRd2latexconcordance = TRUE"concordance"Rconcordance

```

```

encoding
outputEncodingoutputEncoding = ""
outputEncodingiconvRd2latexRd2ex

```

```

\Sexpr

```

<https://developer.r-project.org/parseRd.pdf>

[parse_RdcheckRdfindHTMLlinksRd2txt_optionsmatchConcordance](#)

```

## Simulate rendering of this (installed) page in HTML and text format
Rd <- Rd_db("tools")[["Rd2HTML.Rd"]]

outfile <- tempfile(fileext = ".html")
Rd2HTML(Rd, outfile, package = "tools") |> browseURL()

outfile <- tempfile(fileext = ".txt")
Rd2txt(Rd, outfile, package = "tools") |> file.show()

```

Rd2txt_options

Rd2txt_options(...)

...

Rd2txt

"* "

FALSE\href

TRUE\code

TRUE

Rd2txt

```
saveOpts <- Rd2txt_options()
saveOpts
Rd2txt_options(minIndent = 4)
Rd2txt_options()[["minIndent"]]
Rd2txt_options(saveOpts)
stopifnot(identical(Rd2txt_options(), saveOpts))
```

Rdiff

```
Rdiff(from, to, useDiff = FALSE, forEx = FALSE,  
      nullPointers = TRUE, Log = FALSE)
```

```
fromto  
useDiff      diff  
forEx        -Ex.Rout"--timings"  
nullPointers 0x00000000  
Log
```

```
R CMD BATCHsQuote<environment: 0x12345678>0x00000000useDifffdiff -bdiff -bw-w  
pdf(compress = FALSE)make checkdiffuseDiff = TRUE  
> ## IGNORE_RDIFF_BEGIN> ## IGNORE_RDIFF_END\dontdiff{}
```

```
Logstatusout  
0L
```

```
R CMD RdiffuseDiff = TRUE
```

Rdindex

```
Rdindex(RdFiles, outFile = "", type = NULL,  
        width = 0.9 * getOption("width"), indent = NULL)
```

```
RdFiles  
outFile      ""  
type         NULL\docTypetype"data"datasets  
width  
indent       width/2width/3
```

RdTextFilter

```
RdTextFilter(ifile, encoding = "unknown", keepSpacing = TRUE,  
             drop = character(), keep = character(),  
             macros = file.path(R.home("share"), "Rd", "macros", "system.Rd"))
```

```
ifile          "Rd"parse_Rd  
encoding       parse_Rd  
keepSpacing  
drop  
keep  
macros         parse_Rd
```

```
"TEXT"aspellkeepSpacingFALSE
```

```
\S3method\S4method\command\docType\email\encoding\file\keyword\link\linkS4class  
\method\pkg\vardropkeepkeepc("RCODE", "COMMENT", "VERB")"TEXT"
```

[aspell](#)filter

Rdutils

```
Rd_db(package, dir, lib.loc = NULL, stages = "build")
```

```
package
dir      manpackage
lib.loc  NULLNULLpackage
stages   dir\Sexpr
```

```
Rd_dbparse_Rd\Sexpr
```

```
parse_Rd
```

```
## Build the Rd db for the (installed) base package.
db <- Rd_db("base")

## Keyword metadata per Rd object.
keywords <- lapply(db, tools:::.Rd_get_metadata, "keyword")
## Tabulate the keyword entries.
kw_table <- sort(table(unlist(keywords)))
## The 5 most frequent ones:
rev(kw_table)[1 : 5]
## The "most informative" ones:
kw_table[kw_table == 1]
```

```
## Concept metadata per Rd file.
concepts <- lapply(db, tools:::.Rd_get_metadata, "concept")
## How many files already have \concept metadata?
sum(sapply(concepts, length) > 0)
## How many concept entries altogether?
length(unlist(concepts))
```

read.00Index

```
00IndexINDEXdemo/00Index
```

```
read.00Index(file)
```


file ""fileconnection

"Item""Description"

formatDL

sha256sum

sha256sum(files, bytes)

files

bytes rawbytesfiles

sha256sum

filesfilesfilesNA

bytes

SHA-crypt.txt

md5sum

as.vector(sha256sum(dir(R.home(), pattern = "^COPY", full.names = TRUE)))
sha256sum(bytes=raw())
sha256sum(bytes=charToRaw("abc"))

showNonASCII

<fc>

showNonASCII(x)

showNonASCIIfile(file)

x

file

`xiconv`(sub = "byte")

`iconv`(to = "ASCII")\u2030o/oo

x

```
out <- c(
  "fran\xE7ais: test of showNonASCII():",
  "\\details{",
  "  This is a good line",
  "  This has an \xfcm\aut in it.",
  "  OK again.",
  "}")
f <- tempfile()
cat(out, file = f, sep = "\n")

showNonASCIIfile(f)
unlink(f)
```

startDynamicHelp

startDynamicHelp(start = TRUE)

start

NA

```
127.0.0.1options("help.ports")startDynamicHelpR_DISABLE_HTTPD
options("help.ports")0
startDynamicHelp
options(help_type = "html")
help.start
127.0.0.1
```

```
0
```

```
help.starthelp(help_type = "html")
Rd2HTML
```

SweaveTeXFilter

```
SweaveTeXFilter(ifile, encoding = "unknown")
```

```
ifile
encoding      readLines
```

```
aspellfilter = "Sweave"
```

testInstalledPackage

```
testInstalledPackage(pkg, lib.loc = NULL, outDir = ".",
  types = c("examples", "tests", "vignettes"),
  srcdir = NULL, Ropts = "", ...)

testInstalledPackages(outDir = ".", errorsAreFatal = TRUE,
  scope = c("both", "base", "recommended"),
  types = c("examples", "tests", "vignettes"),
  srcdir = NULL, Ropts = "", ...)

testInstalledBasic(scope = c("basic", "devel", "both", "internet", "all"),
  outDir = file.path(R.home(), "tests"),
  testSrcdir = getTestSrcdir(outDir))
```

standard_package_names()

```
pkg
lib.loc      library
outDir       "."
types
srcdir       .save
Ropts        -d valgrindR CMD BATCH
errorsAreFatal
scope        "both""basic""devel""all""internet"
...          commentDontruncommentDonttest
testSrcdir
```

```
testInstalledPackage{s}()
testsR CMD INSTALL --install-tests
-testsoutDir
testInstalledBasictestSrcdirLC_COLLATELANGUAGEenLC_TIMEC
TEST_MC_COREtestInstalledPackages
make install-tests
diff
```

```
ØL1L
standard_package_names()list
```

```
base          character
recommended    character
```

```

str(stPkgs <- standard_package_names())

## consistency of packageDescription and standard_package_names :
(pNms <- unlist(stPkgs, FALSE))
(prio <- sapply(as.vector(pNms), packageDescription, fields = "Priority"))
stopifnot(identical(unname(prio),
                     sub("[0-9]+$", '', names(pNms))))

```

texi2dvi

latexpdflatexmakeindexbibtex

```

texi2dvi(file, pdf = FALSE, clean = FALSE, quiet = TRUE,
         texi2dvi = getOption("texi2dvi"),
         texinputs = NULL, index = TRUE)

```

```

texi2pdf(file, clean = FALSE, quiet = TRUE,
         texi2dvi = getOption("texi2dvi"),
         texinputs = NULL, index = TRUE)

```

```

file
pdf          TRUEtexi2dvi--pdf
clean        TRUE
quiet        texi2dvi--quiet
texi2dvi     NULL""texi2dvi"NULLtexi2dvisystem2"emulation"
texinputs    NULL
index

```

```

texi2pdftexi2dvi(pdf = TRUE)
Rd2pdf/share/texmfTEXINPUTSSweave.styRd.stytexinputsTEXINPUTS/share/texmf
BIBINPUTSBSTINPUTS
texi2dviR_TEXI2DVICMDTEXI2DVI
"texi2dvi"texify.exetexi2dviNULLtexi2dvi
texi2dvitexi2dvi--pdfpdf = TRUE--quietquiet = TRUE--help--versionlatexmk
hyperrefindex = FALSE

```

```

NULLclean = FALSE

```

```
texi2dvi texi2dvi = "emulation"
texi2dvi 4.8
--max-iterations=20
clean = TRUE
LATEXPDFLATEXMAKEINDEXBIBTEX
```

toHTML

```
toHTML(x, ...)
## S3 method for class 'packageIQR'
toHTML(x, ...)
## S3 method for class 'news_db'
toHTML(x, ...)
```

```
x
...      "packageIQR" "news_db" HTMLheader
```

```
x"packageIQR"
```

[HTMLheader](#)

```
cat(toHTML(demo(package = "base")), sep = "\n")
```

tools-deprecated

[DeprecatedDefunct](#)

toRd

```
toRd(obj, ...)
## S3 method for class 'bibentry'
toRd(obj, style = NULL, ...)
```

```
obj
style      bibentry
...
```

```
bibstylestyle = NULL
```

```
objcharacterbibentry
```

toTitleCase

```
toTitleCase(text)
```

```
text
```

```
text
```

```
toTitleCase("bayesian network modeling and analysis")
toTitleCase("ensemble tool for predictions from species distribution models")
## Treatment after "-":
toTitleCase("small- and large-scale analysis") # lowercase "and"

toTitleCase("a small fox is jumping") # "a Small Fox is .." (the 'a' may change)% i.e. BUG
toTitleCase("is a small fox jumping?") # "Is a Small Fox .." (fine)
## After ":", start a new sentence
toTitleCase("a pangram: the quick brown fox jumps over the lazy dog")
toTitleCase("asking -- 'is a small fox jumping?'") # ".. -- Is a Small ..." (fine)
```

undoc

```
undoc(package, dir, lib.loc = NULL)
```

```
package
dir      manRdata
lib.loc  NULLNULLpackage
```

```
.ArgsEnv.GenericArgsEnvundoc("base")
```

```
"undoc"
print
```

[codocQC](#)

```
undoc("tools")          # Undocumented objects in 'tools'
```

update_PACKAGES

```
PACKAGES
update_PACKAGESwrite\_PACKAGES
```

```
update_PACKAGES(dir = ".", fields = NULL, type = c("source",
  "mac.binary", "win.binary"), verbose.level = as.integer(dryrun),
  latestOnly = TRUE, addFiles = FALSE, rds_compress = "xz",
  strict = TRUE, dryrun = FALSE)
```

```
dir      write\_PACKAGES
fields   write\_PACKAGES
type     write\_PACKAGES
verbose.level dryrunFALSE
latestOnly write\_PACKAGES
addFiles  write\_PACKAGES
rds_compress write\_PACKAGES
strict    PACKAGESTRUE
dryrun     PACKAGESFALSE
```



```

dir_.typeFilePACKAGESPACKAGES
update_PACKAGESwrite_PACKAGES

    type"win.binary"strictTRUEPACKAGES
    PACKAGESdir
    PACKAGESdir
    fieldsNULLPACKAGES

update_PACKAGESPACKAGES
PACKAGESPACKAGESPACKAGESdirPACKAGES
strictTRUEPACKAGESwrite_PACKAGESlatestOnlyTRUE
strictFALSEPACKAGESlatestOnlyTRUE
PackageVersionwrite_PACKAGES
available.packages
PACKAGESPACKAGES
verbose.level01write_PACKAGESverbose    =    FALSEverbose.level2verbose.level1
write_PACKAGESverbose = TRUE

```

```

write_PACKAGEStype == "win.binary"write_PACKAGESupdate_PACKAGES

```

```

## Not run:
write_PACKAGES("c:/myFolder/myRepository") # on Windows
update_PACKAGES("c:/myFolder/myRepository") # on Windows
write_PACKAGES("/pub/RWin/bin/windows/contrib/2.9",
type = "win.binary") # on Linux
update_PACKAGES("/pub/RWin/bin/windows/contrib/2.9",
type = "win.binary") # on Linux

## End(Not run)

```

update_pkg_po

po

```
update_pkg_po(pkgdir, pkg = NULL, version = NULL,  
              pot_make = TRUE, mo_make = TRUE,  
              verbose = getOption("verbose"),  
              mergeOpts = "", copyright, bugs)
```

pkgdir

pkg NULLDESCRIPTION

version NULLDESCRIPTION

pot_makemo_make [logical](#)*.pot*.mo

verbose [logical](#)

mergeOpts msgmerge"--update""--no-wrap"file.path(R.home("po"),
 "Makefile")

copyrightbugs CopyrightReport-Msgid-Bugs-To

po

[xgettext2pot](#)po/R-.pot

poR-.poR-.pot[checkPoFile](#)inst/po

R-en@quot.poinst/po

po/.potsrc/*.{c,cc,cpp,m,mm}po/.potsrc/windows

po.po.pot[checkPoFile](#)inst/po

en@quot.poinst/po

po/.pot

pkg = "base"

gettext-toolsxgettextmsgmergemsgfmtmsginitmsgconv[https://www.stats.ox.ac.uk/
pub/Rtools/goodies/gettext-tools.zip](https://www.stats.ox.ac.uk/pub/Rtools/goodies/gettext-tools.zip)

en@quot

[xgettext2pot](#)

urltools

```
check_package_urls(dir, verbose = FALSE)
parse_URI_reference(x)
```

```
dir
verbose
x
```

```
DESCRIPTIONDescriptionURLBugReportsCITATIONNEWS.md.htmlinst/docpandocREADME.md
NEWS.md
HEADGETHEAD
parse_URI_reference(https://www.rfc-editor.org/rfc/rfc3986)
```

```
check_url_db()check_url_db
parse_URI_reference()schemeauthoritypathqueryfragment
```

```
## Examples from RFC 3986.
parse_URI_reference(c("foo://example.com:8042/over/there?name=ferret#nose",
                      "urn:example:animal:ferret:nose",
                      "mailto:John.Doe@example.com",
                      "tel:+1-816-555-1212"))
```

userdir

```
R_user_dir(package, which = c("data", "config", "cache"))
```

```
package
which
```

<https://specifications.freedesktop.org/basedir-spec>

R_user_dirR

R_USER_DATA_DIRR_USER_CONFIG_DIRR_USER_CACHE_DIRXDG_DATA_HOMEXDG_CONFIG_HOME
XDG_CACHE_HOME

R_user_dir("FOO", "cache")

```
## Create one, platform agnostically, must work if <normal> :  
(Rdb <- R_user_dir("base"))  
if(newD <- !dir.exists(Rdb)) # should work user specifically:  
  newD <- dir.create(Rdb, recursive=TRUE)  
dir(Rdb) # typically empty  
if(newD) unlink(Rdb) # cleaning up  
  
list.files(R_user_dir("grid"), full.names = TRUE)
```

vignetteEngine

[Sweave](#)

```
vignetteEngine(name, weave, tangle, pattern = NULL,  
               package = NULL, aspell = list())
```

name

weave

tangle

pattern NULL

package vignetteEngine

aspell filtercontrol[aspell](#)

weavevignetteEnginenamepackage

weaveNULL

weavetanglefunction(file, ...)...quietencoding[SweaveStangle](#)

weavetangleweave<name><pattern><name>[.](tex|pdf|html).texpdf|latex.pdf|tangle
<name>[.][rRsS]tanglesplit = TRUE<name>.*[.][rRsS]

patternNULL"[.][RrSs](nw|tex)\$"

NULL

name

package

pattern

weave

tangle

[Sweave](#)

```
str(vignetteEngine("Sweave"))
```

vignetteInfo

vignetteInfo(file)

file

[listcharacter](#)

file	basename
title	\VignetteIndexEntry
depends	\VignetteDepends
keywords	\VignetteKeyword
engine	vignetteEngine "utils::Sweave""knitr::knitr"

[package_dependencies](#)

```
gridEx <- system.file("doc", "grid.Rnw", package = "grid")
vi <- vignetteInfo(gridEx)
str(vi)
```

write_PACKAGES

PACKAGESPACKAGES.gzPACKAGES.rds

```
write_PACKAGES(dir = ".", fields = NULL,
               type = c("source", "mac.binary", "win.binary"),
               verbose = FALSE, unpacked = FALSE, subdirs = FALSE,
               latestOnly = TRUE, addFiles = FALSE, rds_compress = "xz",
               validate = FALSE)
```

dir	PACKAGESPACKAGES.gzPACKAGES.rds
fields	PACKAGESPACKAGES.gzPACKAGES.rdsNULL available.packages "Package""Version""Priority""Depends" "Imports""LinkingTo""Suggests""Enhances""OS_type""License" "Archs""File"addFile = TRUE"Path"
type	.tar.{gz,bz2,xz,zstd}.tgz.zip"win.binary""source"
verbose	
unpacked	
subdirs	
latestOnly	
addFiles	FilePACKAGES
rds_compress	PACKAGES.rds saveRDS
validate	DESCRIPTION

```
write_PACKAGESDESCRIPTIONPACKAGESPACKAGES.gzPACKAGES.rdssaveRDS
latestOnly = FALSEfoo_1.0R >= 2.15.0foo_0.9R >= 2.11.0
subdirs != FALSE"Path"PACKAGES
"File"PACKAGESdownload.packagesaddFiles = TRUE
type = "win.binary"unzDESCRIPTIONdirPACKAGESPACKAGES.gzPACKAGES.rds
available.packagesrds_compress = "xz"
```

PACKAGESPACKAGES.gzPACKAGES.rds0

```
.tar.gzDESCRIPTION
type = "win.binary"
```

```
read.dcfwrite.dcfDESCRIPTIONPACKAGESPACKAGES.gzupdate_PACKAGESPACKAGES
PACKAGES.gz
```

```
## Not run:
write_PACKAGES("c:/myFolder/myRepository") # on Windows
write_PACKAGES("/pub/RWin/bin/windows/contrib/2.9",
               type = "win.binary") # on Linux

## End(Not run)
```

xgettext

```
R

xgettext() stopwarningmessagepackageStartupMessagegettextgettextf
xngettext() ngettext

xgettext2pot()xgettext()xngettext()
```

```
xgettext(dir, verbose = FALSE, asCall = TRUE)

xngettext(dir, verbose = FALSE)

xgettext2pot(dir, potFile, name = "R", version, bugs)
```

```
dir          ./R
verbose
asCall       TRUE
potFile      poR-.potdir
nameversionbugs
              versionbugs"bugs.r-project.org"
```

```
\nxgettext()
domain = NAasCallgettextwarningasCall
xgettext2potxgettextxngettextpotFilenggettextmsgfmt
.R/share/R
```

```
xgettext"xgettext"print
xngettext"ngettext"
```

```
update_pkg_po()xgettext2pot()
```

```
## Not run: ## in a source-directory build (not typical!) of R;  
## otherwise, download and unpack the R sources, and replace  
## R.home() by "<my_path_to_source_R>" :  
xgettext(file.path(R.home(), "src", "library", "splines"))
```

```
## End(Not run)
```

```
## Create source package-like <tmp>/R/foo.R and get text from it:  
tmpPkg <- tempdir()  
tmpRDir <- file.path(tmpPkg, "R")  
dir.create(tmpRDir, showWarnings = FALSE)  
fnChar <- paste(sep = "\n",  
  "foo <- function(x) {",  
  "  if (x < -1) stop('too small')",  
  "  # messages unduplicated (not so for ngettext)",  
  "  if (x < -.5) stop('too small')",  
  "  if (x < 0) {",  
  "    warning(",  
  "      'sqrt(x) is', sqrt(as.complex(x)),",  
  "      ', which may be too small'",  
  "    )",  
  "  }",  
  "  # calls with domain=NA are skipped",  
  "  if (x == 0) cat(gettext('x is 0!\n', domain=NA))",  
  "  # gettext strings may be ignored due to 'outer' domain=NA",  
  "  if (x > 10) warning('x is ', gettextf('%.2f', x), domain=NA)",  
  "  # using a custom condition class",  
  "  if (x == 42)",  
  "    stop(errorCondition(gettext('needs Deep Thought'), class='myError'))",  
  "  x",  
  "}")
```

```
writelnLines(fnChar, con = file.path(tmpRDir, "foo.R"))
```

```
## [[1]] : suppressing (tmpfile) name to make example Rdiff-able  
xgettext(tmpPkg, asCall=TRUE)[[1]] # default; shows calls  
xgettext(tmpPkg, asCall=FALSE)[[1]] # doesn't ; but then ' %.2f '
```

```
unlink(tmpRDir, recursive=TRUE)
```


utils

utils-package

library(help = "utils")

<R-core@r-project.org>

adist

```
adist(x, y = NULL, costs = NULL, counts = FALSE, fixed = TRUE,
      partial = !fixed, ignore.case = FALSE, useBytes = FALSE)
```

x	
y	NULLxy
costs	insertionsdeletionssubstitutionsNULL
counts	"counts"
fixed	TRUExpartial = TRUEagrepfixed = FALSE
partial	xyagrep
ignore.case	TRUE
useBytes	TRUE

```
partial    =    FALSEhttps://en.wikipedia.org/wiki/Levenshtein\_distance $O(mn)mn$   
 $O(\max(m,n))$   
https://en.wikipedia.org/wiki/Approximate\_string\_matchingpartial    =    TRUE  
https://github.com/laurikari/treagrep
```

```
xyxy  
countsTRUE"counts"xypartial = FALSE"trafos"MIDSpartial = TRUE"offsets"-1
```

[agrep](#)

```
## Cf. https://en.wikipedia.org/wiki/Levenshtein\_distance  
adist("kitten", "sitting")  
## To see the transformation counts for the Levenshtein distance:  
drop(attr(adist("kitten", "sitting", counts = TRUE), "counts"))  
## To see the transformation sequences:  
attr(adist(c("kitten", "sitting"), counts = TRUE), "trafos")  
  
## Cf. the examples for agrep:  
adist("lasy", "1 lazy 2")  
## For a "partial approximate match" (as used for agrep):  
adist("lasy", "1 lazy 2", partial = TRUE)
```

alarm

```
alarm()
```

```
alarm()"\a"  
flush.console
```

```
alarm()
```

apropos

```
apropos()what  
find()
```

```
apropos(what, where = FALSE, ignore.case = TRUE,  
        dot_internals = FALSE, mode = "any")  
  
find(what, mode = "any", numeric = FALSE, simple.words = TRUE)
```

```
what          findsimple.words = TRUE  
wherenumeric  
ignore.case   TRUE  
dot_internals FALSE  
mode          "any"modemode  
simple.words   TRUEwhat
```

```
mode != "any"mode  
findapropossimple.words == TRUEapropos  
ls.dot_internals
```

```
aproposwhere = TRUE  
findnumeric = TRUE
```

```
glob2rx  
objectshelp.searchsearch
```

```
require(stats)
```

```
## Not run: apropos("lm")  
apropos("GLM") # several  
apropos("GLM", ignore.case = FALSE) # not one  
apropos("lq")
```

```
cor <- 1:pi  
find("cor") #> ".GlobalEnv" "package:stats"
```

```

find("cor", numeric = TRUE) # numbers with these names
find("cor", numeric = TRUE, mode = "function") # only the second one
rm(cor)

## Not run: apropos(".", mode = "list") # includes many datasets

# extraction/replacement methods (need a DOUBLE backslash '\\')
apropos("\\[")

# everything % not diff-able
length(apropos("."))

# those starting with 'pr'
apropos("^pr")

# the 1-letter things
apropos("^. $")
# the 1-2-letter things
apropos("^..?$")
# the 2-to-4 letter things
apropos("^.{2,4}$")
# frequencies of 8-and-more letter things
table(nchar(apropos("^.{8,}$")))

```

aregexec

```

aregexec(pattern, text, max.distance = 0.1, costs = NULL,
          ignore.case = FALSE, fixed = FALSE, useBytes = FALSE)

```

pattern	fixed = FALSE	as.character
text		as.character
max.distance		agrep
costs		agrep
ignore.case	TRUE	
fixed	TRUE	
useBytes	TRUE	

```

aregexec agrep regexec grep
agrep aregexec
agrep adist
pattern text "bytes"

```

```

text - 1 pattern "match.length" - 1

```

regmatches

```
## Cf. the examples for agrep.
x <- c("1 lazy", "1", "1 LAZY")
aregexec("lasy", x, max.distance = 2)
aregexec("(lay)(sy)", x, max.distance = 2)
aregexec("(lay)(sy)", x, max.distance = 2, ignore.case = TRUE)
m <- aregexec("(lay)(sy)", x, max.distance = 2)
regmatches(x, m)
```

arrangeWindows

```
(.Platform$OS.type == "windows")
```

```
arrangeWindows(action, windows, preserve = TRUE, outer = FALSE)
```

action	c("vertical", "horizontal", "cascade", "minimize", "restore") "vertical"
windows	listgetWindowsHandles()
preserve	TRUE
outer	TRUE

```
"vertical"
```

```
"horizontal"
```

```
"cascade"
```

```
"minimize"
```

```
"restore"
```

```
windows
```

```
windowsgetWindowsHandles().arrangeWindowsDefaultsgetWindowsHandles
```

```
action = "restore"windowsminimized = TRUEgetWindowsHandles
```

```
outer = TRUEwindows
```

getWindowsHandles

```
## Not run: ## Only available on Windows :
arrangeWindows("v")
# This default is useful only in SDI mode: it will tile any Firefox window
# along with the R windows
.arrangeWindowsDefaults <- list(c("R", "all"), pattern = c("", "Firefox"))
arrangeWindows("v")

## End(Not run)
```

askYesNo

askYesNo

```
askYesNo(msg, default = TRUE,
         prompts = getOption("askYesNo", gettext(c("Yes", "No", "Cancel"))),
         ...)
```

```
msg
default
prompts      TRUEFALSENA/
...
```

```
askYesNodefault
promptsfn(msg = msg, default = default, prompts = prompts, ...)
utils::askYesNoWinDialog
"Y/N/C"prompts
```

TRUEFALSENA

readline

```
if (interactive())
  askYesNo("Do you want to use askYesNo?")
```

aspell

```
aspell(files, filter, control = list(), encoding = "unknown",
       program = NULL, dictionaries = character())
```

files

filter ifileencodingNULL

control

encoding

program NULLaspellhunspellispell

dictionaries share/dictionaries.rds

```
-ahttp://aspell.nethttps://hunspell.github.io/https://www.cs.hmc.edu/~geoff/ispell.html
```

```
mingw-w64-x86_64-aspellaspellaspell()
```

```
"Rd"RdTextFilterignore"Sweave"SweaveTeXFilter"R""pot""dcf""md"
```

```
"R"messagewarningstoppackageStartupMessagegettextgettextfngettextfmtmsg1msg2
```

```
"pot".potignore"[ \t]'[^']*'[ \t[:punct:]]"
```

```
"dcf"keepTitleDescription
```

```
"md".md.Rmd
```

```
aspellindentverbose
```

```
.rdssaveRDS-p
```

aspell


```
## Not run:
## To check all Rd files in a directory, (additionally) skipping the
## \references sections.
files <- Sys.glob("*.Rd")
aspell(files, filter = list("Rd", drop = "\\references"))

## To check all Sweave files
files <- Sys.glob(c("*.Rnw", "*.Snw", "*.rnw", "*.snw"))
aspell(files, filter = "Sweave", control = "-t")

## To check all Texinfo files (Aspell only)
files <- Sys.glob("*.texi")
aspell(files, control = "--mode=texinfo")

## End(Not run)

## List the available R system dictionaries.
Sys.glob(file.path(R.home("share"), "dictionaries", "*.rds"))
```

aspell-utils

```
aspell_package_Rd_files(dir,
                        drop = c("\\abbr", "\\acronym",
                                "\\author", "\\references"),
                        control = list(), program = NULL,
                        dictionaries = character())
aspell_package_vignettes(dir,
                        control = list(), program = NULL,
                        dictionaries = character())
aspell_package_R_files(dir, ignore = character(), control = list(),
                        program = NULL, dictionaries = character())
aspell_package_C_files(dir, ignore = character(), control = list(),
                        program = NULL, dictionaries = character())

aspell_write_personal_dictionary_file(x, out, language = "en",
                                     program = NULL)
```

```
dir
drop      RdTextFilter
control
program   NULLaspellhunspellispell
dictionaries  aspell
ignore
x          aspell()
out
language
```

```

aspell_package_Rd_files aspell_package_vignettes aspell_package_R_files
aspell_package_C_files dir
aspell
po/.potdir
\Sexpr\citep\code\pkg\proglang\samp--add-tex-command control\mycmd
--add-tex-command='mycmd op'
controlprogramdictionariesdropignoredefaults.R. aspell dir
vignettes <- list(control = "--add-tex-command='mycmd op'")
Rd_files R_files C_files
--master=en_US--add-extra-dicts=en_GB-d en_US,en_GB
aspell_write_personal_dictionary_file.rds saveRDS. aspell

```

aspell

available.packages

available.packages

```

available.packages(contriburl = contrib.url(repos, type), method,
  fields = getOption("available_packages_fields"),
  type = getOption("pkgType"), filters = NULL,
  repos = getOption("repos"),
  ignore_repo_cache = FALSE, max_repo_cache_age,
  cache_user_dir =
    str2logical(Sys.getenv("R_PACKAGES_CACHE_USER_DIR",
      FALSE)),
  quiet = TRUE, verbose = FALSE, ...)

```

contriburl	contribcontrib
method	download.file
type	install.packages
	type = "both"
fields	PACKAGESNULLNA
filters	NULL
repos	
ignore_repo_cache	
max_repo_cache_age	

cache_user_dir	logicalR_user_dir ("base", "cache") tempdir ()
quiet	download.file ()
verbose	
...	download.file ()

```

file://ignore_repo_cachetempdir()

repos_http%3a%2f%2fcran.r-project.org%2fsrc%2fcontrib.rds

max_repo_cache_ageR_AVAILABLE_PACKAGES_CACHE_CONTROL_MAX_AGE3600

filtersNULLinstall.packagesgetOption("available_packages_filters")
c("R_version", "OS_type", "subarch", "duplicates")NULL

"R_version"
"OS_type"
"subarch"
"duplicates" contriburl
"license/FOSS" https://en.wikipedia.org/wiki/FOSS

"license/restricts_use"
"CRAN" "duplicates"add = TRUE

filtersadd = TRUE
available.packages
add = TRUE

"Package""Version""Priority""Depends""Imports""LinkingTo""Suggests""Enhances"
"File""Repository"fields
"OS_type""License""License_is_FOSS""License_restricts_use""Archs""MD5sum"
"NeedsCompilation"install.packages

packageStatusupdate.packagesinstall.packagesdownload.packagescontrib.url

## Count package licenses
db <- available.packages(repos = findCRANmirror("web"), filters = "duplicates")
table(db[, "License"])

## Use custom filter function to only keep recommended packages
## which do not require compilation
available.packages(repos = findCRANmirror("web"),
  filters = list(
    add = TRUE,
    function (db) db[db[, "Priority"] %in% "recommended" &
      db[, "NeedsCompilation"] == "no", ]
  ))

## Not run:

```

```

## Restrict install.packages() (etc) to known-to-be-FOSS packages
options(available_packages_filters =
  c("R_version", "OS_type", "subarch", "duplicates", "license/FOSS"))
## or
options(available_packages_filters = list(add = TRUE, "license/FOSS"))

## Give priority to released versions on CRAN, rather than development
## versions on R-Forge etc.
options(available_packages_filters =
  c("R_version", "OS_type", "subarch", "CRAN", "duplicates"))

## End(Not run)

```

BATCH

infile

R CMD BATCH [options] infile [outfile]

infile

options infile-----restore --save --no-readline--no-readline

outfile infile.R.Rout

R CMD BATCH --help

options(echo = FALSE)infile--no-echo

infile

proc.time()q(runLast = FALSE)--no-timing

R_BATCH_OPTIONSOptions

Splus BATCH

R CMD BATCH [options] infile [outfile] &

bibentry

```
bibentry(bibtype, textVersion = NULL, header = NULL, footer = NULL,  
         key = NULL, ..., other = list(),  
         mheader = NULL, mfooter = NULL)
```

```
## S3 method for class 'bibentry'  
print(x, style = "text", .bibstyle,  
      bibtex = length(x) <= getOption("citation.bibtex.max", 1),  
      ...)
```

```
## S3 method for class 'bibentry'  
format(x, style = "text", .bibstyle = NULL,  
       bibtex = length(x) <= 1,  
       citMsg = missing(bibtex),  
       sort = FALSE, macros = NULL, ...)
```

```
## S3 method for class 'bibentry'  
sort(x, decreasing = FALSE, .bibstyle = NULL, drop = FALSE, ...)
```

```
## S3 method for class 'citation'  
print(x, style = "citation", ...)  
## S3 method for class 'citation'  
format(x, style = "citation", ...)
```

```
## S3 method for class 'bibentry'  
toBibtex(object, escape = FALSE, ...)
```

bibtype

textVersion format(x, style = "text")textVersion

header

footer

key

... bibentry=
 print()format()
 citationbibentry
 toBibtex()

other ...bibentry

mheader

mfooter

x "bibentry"

```

style
decreasing      order
.bibstyle       bibstyle
bibtex          logicalstyle = "citation"print()getOption("citation.bibtex.max")
                options(citation.bibtex.max = 0)
citMsg          logicalbibtexstyle = "citation"
sort            bibstyle(.bibstyle)$sortKeys(x)
macros
drop            x[ ..., drop=drop]sort()
object          "bibentry"
escape

```

```

bibentryc()bibentry
printformat"text""bibtex""citation""html""latex""R"textVersion"textVersion"
"text""html""latex".bibstylebibstyle\boldmacrosloadRdMacros"latex"Rd.sty
\usepackage{Rd}texi2pdf
headerfootermheadermfooter
bibentry()CITATIONc()collapse=FALSETRUEformat()
NULL
toBibtex
transform

```

```

bibentry"bibentry"

```

```

bibentry"bibentry"

```

...bibentry

[personas.person](#)

https://en.wikipedia.org/wiki/Digital_Object_Identifier
author

doi

[person](#)

```
## R reference
rref <- bibentry(
  bibtype = "Manual",
  title = "R: A Language and Environment for Statistical Computing",
  author = person("R Core Team"),
  organization = "R Foundation for Statistical Computing",
  address = "Vienna, Austria",
  year = 2014,
  url = "https://www.R-project.org/")

## Different printing styles
print(rref)
print(rref, style = "bibtex")
print(rref, style = "citation")
print(rref, style = "html")
print(rref, style = "latex")
print(rref, style = "R")

## References for boot package and associated book
bref <- c(
  bibentry(
    bibtype = "Manual",
```

```

    title = "boot: Bootstrap R (S-PLUS) Functions",
    author = c(
      person("Angelo", "Canty", role = "aut",
        comment = "S original"),
      person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
        comment = "R port, author of parallel support",
        email = "ripley@stats.ox.ac.uk")
    ),
    year = "2012",
    note = "R package version 1.3-4",
    url = "https://CRAN.R-project.org/package=boot",
    key = "boot-package"
  ),

  bibentry(
    bibtype = "Book",
    title = "Bootstrap Methods and Their Applications",
    author = as.person("Anthony C. Davison [aut], David V. Hinkley [aut]"),
    year = "1997",
    publisher = "Cambridge University Press",
    address = "Cambridge",
    isbn = "0-521-57391-2",
    url = "http://statwww.epfl.ch/davison/BMA/",
    key = "boot-book"
  )
)

## Combining and subsetting
c(rref, bref)
bref[2]
bref["boot-book"]

## Extracting fields
bref$author
bref[1]$author
bref[1]$author[2]$email

## Field names are case-insensitive
rref$Year
rref$Year <- R.version$year
stopifnot(identical(rref$year, R.version$year))

## Convert to BibTeX
toBibtex(bref)

## Transform
transform(rref, address = paste0(address, ", Europe"))

## BibTeX reminder message (in case of >= 2 refs):
print(bref, style = "citation")

## Format in R style
## One bibentry() call for each bibentry:
writeLines(paste(format(bref, "R"), collapse = "\n\n"))
## One collapsed call:
writeLines(format(bref, "R", collapse = TRUE))

```

browseEnv

```
browseEnvsys.frame()
```

```
browseEnv(envir = .GlobalEnv, pattern,  
          excludepatt = "^last\\.warning",  
          html = .Platform$GUI != "AQUA",  
          expanded = TRUE, properties = NULL,  
          main = NULL, debugMe = FALSE)
```

envir	environment
pattern	ls()
excludepatt	
html	R.app
expanded	FALSEhtmlFALSE
properties	NULL
main	NULL
debugMe	

R.app

<https://www.bioconductor.org>

strls

```
if(interactive()) {  
  ## create some interesting objects :  
  ofa <- ordered(4:1)  
  ex1 <- expression(1+ 0:9)  
  ex3 <- expression(u, v, 1+ 0:9)  
  example(factor, echo = FALSE)  
  example(table, echo = FALSE)  
  example(ftable, echo = FALSE)  
  example(lm, echo = FALSE, ask = FALSE)  
  example(str, echo = FALSE)  
  
  ## and browse them:  
  browseEnv()  
  
  ## a (simple) function's environment:  
  af12 <- approxfun(1:2, 1:2, method = "const")  
  browseEnv(envir = environment(af12))  
}
```

browseURL

```
browseURL(url, browser = getOption("browser"),
          encodeIfNeeded = FALSE)
```

```
url
```

```
browser
```

```
NULL
```

```
encodeIfNeeded URLencodebrowserfile://http://
```

```
"browser"R_BROWSER/etc/RenvironStartup"false"
```

```
  "browser"openxdg-open
```

```
  browser"-remote openURL(...)"
```

```
  "-remote"DISPLAY
```

```
  urlURLencode
```

```
  browser = "false"
```

```
  url
```

```
"browser"R_BROWSERNULL"false"
```

```
  :|\\
```

```
  browser = "false"
```

```
http://https://ftp://mailto:ftp://
```

```
file://
```

```
## Not run:
```

```
## for KDE users who want to open files in a new tab
```

```
options(browser = "kfmclient newTab")
```

```
browseURL("https://www.r-project.org")
```

```
## On Windows-only, something like
```

```
browseURL("file://d:/R/R-2.5.1/doc/html/index.html",
```

```
          browser = "C:/Program Files/Mozilla Firefox/firefox.exe")
```

```
## End(Not run)
```

browseVignettes

```
browseVignettes(package = NULL, lib.loc = NULL, all = TRUE)
```

```
## S3 method for class 'browseVignettes'  
print(x, ...)
```

```
package      NULLall  
lib.loc       NULLNULL  
all           TRUElib.locFALSE  
x             browseVignettes  
...           print
```

browseVignettes[browseURL](#)

[browseURLvignette](#)

```
## List vignettes from all *attached* packages  
browseVignettes(all = FALSE)
```

```
## List vignettes from a specific package  
browseVignettes("grid")
```

bug.report

```
bug.report(subject = "", address,  
            file = "R.bug.report", package = NULL, lib.loc = NULL,  
            ...)
```

```
subject  
address  
file  
package  
lib.loc      NULLNULL  
...          methodccaddresscreate.post
```

```
packageNULLhttps://bugs.r-project.org/  
packageDESCRIPTIONBugReportsDESCRIPTIONbrowseURLcreate.postBugReportsContact
```

```
r-devel@r-project.org
```

```
version
```

```
data.frame(x, y, z, monday, tuesday)data.frame()data.frame()  
[data.frame()]
```

```
--vanilla  
bug.report()https://bugs.r-project.org/  
package
```

```
help.requestbug.report  
create.post  
sessionInfo\(\)
```

```
capture.output
```

```
sinkwithattach
```

```
capture.output(..., file = NULL, append = FALSE,  
               type = c("output", "message"), split = FALSE)
```

```

...
file          NULL
append        file
typesplit     sink()

sink(<file connection>)dotscloseAllConnections()
filefile = NULL
stderr()messagewarningstoptype = "message"

file = NULLNULL

sinktextConnection

require(stats)
glmout <- capture.output(summary(glm(case ~ spontaneous+induced,
                                     data = infert, family = binomial()))))
glmout[1:5]
capture.output(1+1, 2+2)
capture.output({1+1; 2+2})

## Not run: ## on Unix-alike with a2ps available
op <- options(useFancyQuotes=FALSE)
pdf <- pipe("a2ps -o - | ps2pdf - tempout.pdf", "w")
capture.output(example(glm), file = pdf)
close(pdf); options(op) ; system("evince tempout.pdf &")

## End(Not run)

```

changedFiles

```

fileSnapshotchangedFiles

fileSnapshot(path = ".", file.info = TRUE, timestamp = NULL,
             md5sum = FALSE, digest = NULL, full.names = length(path) > 1,
             ...)

changedFiles(before, after, path = before$path, timestamp = before$timestamp,
             check.file.info = c("size", "isdir", "mode", "mtime"),
             md5sum = before$md5sum, digest = before$digest,
             full.names = before$full.names, ...)

## S3 method for class 'fileSnapshot'
print(x, verbose = FALSE, ...)

## S3 method for class 'changedFiles'
print(x, verbose = FALSE, ...)

```

```

path
file.info      file.info
timestamp      NULL
md5sum
digest         NULLfunction(filename)
full.names     list.filesTRUElength(path) > 1
...            list.files
beforeafter    fileSnapshotafterbefore
check.file.info
               file.info

x
verbose

```

```

fileSnapshotlist.filesfile.infomd5sumdigest
changedFiles
timestampfileSnapshotchangedFilesfile_testbeforeafterafterafter
check.file.infofile.info
md5sumTRUEfileSnapshottools::md5sumchangedFilesdigest

```

```
fileSnapshot"fileSnapshot"
```

```

info
path          path
timestampfile.infomd5sumdigestfull.names

```

```
args          ...list.files
```

```
changedFiles"changedFiles"
```

```
addeddeletedchangedunchanged
```

```
changes      TRUE
```

```
printprint"fileSnapshot""changedFiles"addeddeletedchangedchangesTRUE
```

```
file.infofile_testmd5sum
```

```

# Create some files in a temporary directory
dir <- tempfile()
dir.create(dir)
writeBin(1L, file.path(dir, "file1"))
writeBin(2L, file.path(dir, "file2"))
dir.create(file.path(dir, "dir"))

# Take a snapshot
snapshot <- fileSnapshot(dir, timestamp = tempfile("timestamp"), md5sum=TRUE)

# Change one of the files.
writeBin(3L:4L, file.path(dir, "file2"))

# Display the detected changes. We may or may not see mtime change...
changedFiles(snapshot)
changedFiles(snapshot)$changes

```

charClass

```
charClass(x, class)
```

```

x
class

```

```

"alnum"
"alpha"
"blank"
"cntrl"
"digit" 0-9
"graph"
"lower"
"print"
"punct"
"space"
"upper"
"xdigit" 0-9A-fa-f

```

```

print.default
xEncoding
charClassgrep1("[[:print:]]", intToUtf8(x))

```

x

"digit"

maniswctypewctype

```
x <- c(48:70, 32, 0xa0) # Last is non-breaking space
cl <- c("alnum", "alpha", "blank", "digit", "graph", "punct", "upper", "xdigit")
X <- lapply(cl, function(y) charClass(x,y)); names(X) <- cl
X <- as.data.frame(X); row.names(X) <- sQuote(intToUtf8(x, multiple = TRUE))
X

charClass("ABC123", "alpha")
## Some accented capital Greek characters
(x <- "\u0386\u0388\u0389")
charClass(x, "upper")

## How many printable characters are there? (Around 280,000 in Unicode 13.)
## There are 2^21-1 possible Unicode points (most not yet assigned).
pr <- charClass(1:0xffffffff, "print")
table(pr)
```

choose.dir

choose.dir(default = "", caption = "Select folder")

default

caption

default = ""

NA

[choose.files](#)[file.choose](#)

```
if (interactive() && .Platform$OS.type == "windows")
  choose.dir(getwd(), "Choose a suitable folder")
```

[choose.files](#)

```
choose.files(default = "", caption = "Select files",
             multi = TRUE, filters = Filters,
             index = nrow(Filters))
```

Filters

```
default
caption
multi
filters
index
```

```
file.choosechoose.filesfile.choosechoose.dir
filterschoose.filesn2index
Filters
"c:\\*.*"default
```

[file.choose](#)[choose.dir](#)

[Sys.globlist.files](#)

```
if (interactive() && .Platform$OS.type == "windows")
  choose.files(filters = Filters[c("zip", "All"),])
```

chooseBioCmirror

```
chooseBioCmirror(graphics = getOption("menu.graphics"), ind = NULL,  
                  local.only = FALSE)
```

graphics menu

ind

local.only

```
"BioC_mirror"setRepositoriesNULLhttps://bioconductor.org
```

```
ind/doc/BioC_mirrors.csv
```

```
options("BioC_mirror")
```

```
setRepositorieschooseCRANmirror
```

chooseCRANmirror

```
chooseCRANmirror(graphics = getOption("menu.graphics"), ind = NULL,  
                  local.only = FALSE)
```

```
getCRANmirrors(all = FALSE, local.only = FALSE)
```

graphics menu

ind

all

local.only

```

/doc/CRAN_mirrors.csv
chooseCRANmirrorcontrib.urloptions("repos")
ssh
indlocal.only = TRUE/doc/CRAN_mirrors.csv

```

```

chooseCRANmirror()options("repos")
getCRANmirrors()

```

```

setRepositoriesfindCRANmirrorchooseBioCmirrorcontrib.url

```

citation

```

citation(package = "base", lib.loc = NULL, auto = NULL)

```

```

readCitationFile(file, meta = NULL)
citHeader(...)
citFooter(...)

```

```

package
lib.loc      packageNULLNULL
auto        DESCRIPTIONNULLCITATION"packageDescription"
file
meta        packageDescriptionNULL
...         paste

```

```

citation()
citation()packageautoCITATIONDESCRIPTIONauto = NULLCITATIONreadCitationFilemeta
packageDescription(package, lib.loc)auto = TRUE

```

```

Authors@RDESCRIPTIONperson
citation()options("citation.bibtex.max")toBibtex()
CITATIONinsteadCitationFile()citation()CITATIONsource()"bibentry"bibentry()
citHeader()citFooter()CITATIONcitation(auto = meta)
readCitationFileEncodingmeta

```

```
"citation""bibentry"printformat
citHeadercitFooter"bibentry"
```

`bibentry`

```
## the basic R reference
citation()

## extract the BibTeX entry from the return value
x <- citation()
toBibtex(x)

## references for a package
citation("lattice")
citation("lattice", auto = TRUE) # request the Manual-type reference
citation("foreign")

## a CITATION file with more than one bibentry:
file.show(system.file("CITATION", package="mgcv"))
cm <- citation("mgcv")
cm # header, text references, plus "reminder" about getting BibTeX
print(cm, bibtex = TRUE) # each showing its bibtex code

## a CITATION file including citation(auto = meta)
file.show(system.file("CITATION", package="nlme"))
citation("nlme")
```

`cite`

```
bibentrycite()cite()bibstyleciteNatbib()citeNatbib()natbib
```

```
cite(keys, bib, ...)
citeNatbib(keys, bib, textual = FALSE, before = NULL, after = NULL,
            mode = c("authoryear", "numbers", "super"),
            abbreviate = TRUE, longnamesfirst = TRUE,
            bibpunct = c("(", ")", ";", "a", "", ",", previous))
```

```
keys
bib      "bibentry"
...      cite()
textual  \citet
before
```

```

after
mode
abbreviate
longnamesfirst abbreviate == TRUE
bibpunct      natbib
previous      abbreviate == TRUElongnamesfirst == TRUE

```

```

natbibbibpunct
bibpunct

```

```

"n"s"

```

```

modebibpunct[4]mode
citeNatbibcitebibstyle

```

```

## R reference
rref <- bibentry(
  bibtype = "Manual",
  title = "R: A Language and Environment for Statistical Computing",
  author = person("R Core Team"),
  organization = "R Foundation for Statistical Computing",
  address = "Vienna, Austria",
  year = 2013,
  url = "https://www.R-project.org/",
  key = "R")

## References for boot package and associated book
bref <- c(
  bibentry(
    bibtype = "Manual",
    title = "boot: Bootstrap R (S-PLUS) Functions",
    author = c(
      person("Angelo", "Canty", role = "aut",
        comment = "S original"),
      person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
        comment = "R port, author of parallel support",
        email = "ripley@stats.ox.ac.uk")
    )
  )
)

```

```

    ),
    year = "2012",
    note = "R package version 1.3-4",
    url = "https://CRAN.R-project.org/package=boot",
    key = "boot-package"
  ),

  bibentry(
    bibtype = "Book",
    title = "Bootstrap Methods and Their Applications",
    author = as.person("Anthony C. Davison [aut], David V. Hinkley [aut]"),
    year = "1997",
    publisher = "Cambridge University Press",
    address = "Cambridge",
    isbn = "0-521-57391-2",
    url = "http://statwww.epfl.ch/davison/BMA/",
    key = "boot-book"
  )
)

## Combine and cite
refs <- c(rref, bref)
cite("R, boot-package", refs)

## Cite numerically
savestyle <- tools::getBibstyle()
tools::bibstyle("JSSnumbered", .init = TRUE,
  fmtPrefix = function(paper) paste0("[", paper$.index, "]"),
  cite = function(key, bib, ...)
    citeNatbib(key, bib, mode = "numbers",
      bibpunct = c("[", "]", ";", "n", "", ",", "...")
    )
)
cite("R, boot-package", refs, textual = TRUE)
refs

## restore the old style
tools::bibstyle(savestyle, .default = TRUE)

```

citEntry

[bibentry](#)

citEntry(entry, textVersion = NULL, header = NULL, footer = NULL, ...)

entry	bibentry
textVersion	
header	
footer	
...	citEntry= bibentry

citEntry"bibentry"

[citation](#)CITATION**[bibentry](#)**

clipboard

```
getClipboardFormats(numeric = FALSE)
readClipboard(format = 13, raw = FALSE)
writeClipboard(str, format = 13)
```

numeric

format

raw

str

<https://learn.microsoft.com/en-gb/windows/desktop/dataxchg/clipboard-formats>

```
raw = TRUEraw = FALSECF_LOCALECF_LOCALE
writeClipboard
```

getClipboardFormats
readClipboarddrawTRUENULL
writeClipboard

file

close.socket

close.socket(socket, ...)

socket socket
...

make.socketread.socket
capabilities("sockets")

combn

```
xmxseq(x)mFUNNULLarraymatrix...FUN
```

```
combn(x, m, FUN = NULL, simplify = TRUE, ...)
```

```
x          nx <- seq_len(n)
m
FUN        NULLm
simplify   arraymatrixlistsimplify = TRUEFUN()FUN(u)
...        FUN
```

```
x
```

```
listarraysimplifydim(combn(n, m)) == c(m, choose(n, m))
```

```
<stvjc@channing.harvard.edu>simplify = TRUEcombn(5,5)
```

[chooseexpand.grid](#)

```
combn(letters[1:4], 2)
(m <- combn(10, 5, min)) # minimum value in each combination
mm <- combn(15, 6, function(x) matrix(x, 2, 3))
stopifnot(round(choose(10, 5)) == length(m), is.array(m), # 1-dimensional
           c(2,3, round(choose(15, 6))) == dim(mm))

## Different way of encoding points:
combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate, nbins = 4)

## Compute support points and (scaled) probabilities for a
## Multivariate-Hypergeometric(n = 3, N = c(4,3,2,1)) p.f.:
# table.mat(t(combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate, nbins = 4)))

## Assuring the identity
for(n in 1:7)
  for(m in 0:n) stopifnot(is.array(cc <- combn(n, m)),
                        dim(cc) == c(m, choose(n, m)),
                        identical(cc, combn(n, m, identity)) || m == 1)
```

compareVersion

compareVersion(a, b)

ab

x.y-zxyzx

0-1b1astrcmp

[package_versionlibrarypackageStatus](#)

compareVersion("1.0", "1.0-1")
compareVersion("7.2-0", "7.1-12")

COMPILE

R CMD SHLIBR CMD LINK

R CMD COMPILE [options] srcfiles

srcfiles .c.cc.cpp.m.mm.M.f.f90.f95
options

R CMD SHLIBR CMD SHLIB

make.o

COMPILER-devel

[LINKSHLIBdyn.load](#)

[RShowDoc\("R-admin"\)](#)

`contrib.url`

`contrib.urlrepos`

`contrib.url(repos, type = getOption("pkgType"))`

`repos`

`type` [install.packages](#)

`type = "both"`

`repos`

[setRepositoriesoptions\("repos"\)repos](#)

[available.packagesdownload.packagesinstall.packages](#)

`count.fields`

`count.fieldssepfile`

`count.fields(file, sep = "", quote = "\"\"", skip = 0,
 blank.lines.skip = TRUE, comment.char = "#")`

`file` [connection](#)

`sep`

`quote`

`skip`

`blank.lines.skip`

`TRUE`

`comment.char`

[read.table](#)

[scan](#)

[scan](#)`count.fieldsNA`

[read.table](#)

```
fil <- tempfile()
cat("NAME", "1:John", "2:Paul", file = fil, sep = "\n")
count.fields(fil, sep = ":")
unlink(fil)
```

create.post

[bug.reporthelp.request](#)

```
create.post(instructions = character(), description = "post",
            subject = "",
            method = getOption("mailer"),
            address = "the relevant mailing list",
            ccaddress = getOption("ccaddress", ""),
            filename = "R.post", info = character())
```

```
instructions
description
subject      "mailx"
method       "none""mailto""gnudoit""ess""mailx"
address
ccaddress    "mailx""mailto"ccaddress = ""
filename     "none"
info
```

```
method
none file.editfile
mailto
      xdg-openR_BROWSER
```

```
mailx file.editmailx
gnudoit gnudoitsubject
ess stdout
```

NULL

[bug.reporthelp.request](#)

data

```
data(..., list = character(), package = NULL, lib.loc = NULL,
      verbose = getOption("verbose"), envir = .GlobalEnv,
      overwrite = TRUE)
```

```
...
```

```
list
```

```
package      NULL
```

```
data
```

```
lib.loc      NULLNULL
```

```
verbose      TRUE
```

```
envir
```

```
overwrite    envir
```

```
.R.rsource()datautils::data
```

```
.RData.rdata.rdaload()
```

```
.tab.txt.TXTread.table(..., header = TRUE, as.is=FALSE)
```

```
.csv.CSVread.table(..., header = TRUE, sep = ";", as.is=FALSE)
```

```
.txt.tab.csv.gz.bz2.xz
```

```
list
```

```
.R.r.RData.rda
```

```
dataMetadatalist_files_with_type"packageIQR"beaver1      (beavers)beaver1
```

```
data(beavers)
```

```
lib.locpackageNULLdata
```

```
lib.loc = NULLpackage.libPaths
```

```
lib.locNULL
```

```
datapackage = character(0)lib.loc = NULL
```

```
"packageIQR"
```

```
data()
data().GlobalEnv

dataenvirdata(..., envir = environment())
```

```
R/sysdata.rda
mytable::survival::survexp.us
```

```
envirdata("foo")foo.Random.seed
```

```
.Rmydata.txtmydata.Rmydata.txttransform().R
```

```
helpsave.rda
data
```

```
require(utils)
data() # list all available data sets
try(data(package = "rpart"), silent = TRUE) # list the data sets in the rpart package
data(USArrests, "VADeaths") # load the data sets 'USArrests' and 'VADeaths'
## Not run: ## Alternatively
ds <- c("USArrests", "VADeaths"); data(list = ds)
## End(Not run)
help(USArrests) # give information on data set 'USArrests'
```

dataentry

```
data.entry(..., Modes = NULL, Names = NULL)
dataentry(data, modes)
de(..., Modes = list(), Names = NULL)
```

```
...
Modes
Names
data
modes          datalist()
```

```
data.entrydede.ncolsde.setupde.restoreXdataentryXdataentry
dataentry
de.ncolsdata.entryde.setupde.restore
```

```
dedataentrydata.entry
```

```
R_dataentryforegroundbackgroundgeometry
```

```
-.eENA><
NA
NA
var5
Coppaste
```

```
de.cellwidth
```

```
viediteditdataentry
```

```
# call data entry with variables x and y
## Not run: data.entry(x, y)
```

```
debugcall
```

```
debugcall(call, once = FALSE)
undebugcall(call)
```

```
call
once          TRUEdebugonceFALSE
```

debugcallcallisS3stdGeneric

debugcall

callsubstitute

debug

```
## Not run:  
## Evaluate call after setting debugging  
##  
f <- factor(1:10)  
res <- eval(debugcall(summary(f)))  
  
## End(Not run)
```

debugger

```
dump.frames(dumpto = "last.dump", to.file = FALSE,  
            include.GlobalEnv = FALSE)  
debugger(dump = last.dump)  
  
limitedLabels(value, maxwidth = getOption("width") - 5L)
```

```
dumpto  
to.file  
include.GlobalEnv  
            .GlobalEnvsys.frames()  
dump        dump.frames  
value       listcall  
maxwidth    limitedLabels()
```

```
errordump.frameslast.dumpsaveterrmessage  
dumpto.rda  
"dump.frames"debuggerbrowser...  
dump.frames  
limitedLabels(v)listsrcrefstrtrim()maxwidth
```


NULL

`sys.parentenvironment`debugger

`browser`Browse

`optionserrorrecover`debugger

```
## Not run:
options(error = quote(dump.frames("testdump", TRUE)))
```

```
f <- function() {
  g <- function() stop("test dump.frames")
  g()
}
f() # will generate a dump on file "testdump.rda"
options(error = NULL)
```

```
## possibly in another R session
load("testdump.rda")
debugger(testdump)
Available environments had calls:
1: f()
2: g()
3: stop("test dump.frames")
```

Enter an environment number, or 0 to exit

Selection: 1

Browsing in the environment with call:

f()

Called from: debugger.look(ind)

Browse[1]> ls()

[1] "g"

Browse[1]> g

function() stop("test dump.frames")

<environment: 759818>

Browse[1]>

Available environments had calls:

1: f()

2: g()

3: stop("test dump.frames")

Enter an environment number, or 0 to exit

Selection: 0

```
## A possible setting for non-interactive sessions
options(error = quote({dump.frames(to.file = TRUE); q(status = 1)}))

## End(Not run)
```

demo

```
demodemo()
```

```
demo(topic, package = NULL, lib.loc = NULL,
      character.only = FALSE, verbose = getOption("verbose"),
      type = c("console", "html"), echo = TRUE,
      ask = getOption("demo.ask"),
      encoding = getOption("encoding"))
```

```
topic          character.onlyFALSETRUE
package        NULL
lib.loc        NULLNULL
character.only TRUEtopic
verbose        TRUE
type           knitrlib.loc
echo           TRUE
ask            "default"devAskNewPage(ask = TRUE)"default"echo == TRUETRUE
encoding       source
```

```
demotype = "console""packageIQR"
```

```
sourcedevAskNewPagedemoexample
```

```
demo() # for attached packages
```

```
## All available demos:
demo(package = .packages(all.available = TRUE))
```

```
## Display a demo, pausing between pages
demo(lm.glm, package = "stats", ask = TRUE)
```

```
## Display it without pausing
demo(lm.glm, package = "stats", ask = FALSE)
```

```
## Not run:
ch <- "scoping"
demo(ch, character = TRUE)

## End(Not run)

## Find the location of a demo
system.file("demo", "lm.glm.R", package = "stats")
```

DLL.version

DLL.version(path)

path

NULL

```
if(.Platform$OS.type == "windows") withAutoprint({
  DLL.version(file.path(R.home("bin"), "R.dll"))
  DLL.version(file.path(R.home(), "library/stats/libs", .Platform$r_arch, "stats.dll"))
})
```

download.file

```
download.file(url, destfile, method, quiet = FALSE, mode = "w",
  cacheOK = TRUE,
  extra = getOption("download.file.extra"),
  headers = NULL, ...)
```

```

url          character"libcurl"
destfile     url
method       "internal""libcurl""wget""curl""wininet""auto"
             "download.file.method"options()
quiet        TRUE
mode         "w""wb""a""ab""wget""curl""wb"
cacheOK      0
extra        "wget""curl"
headers      User-AgentHTTPUserAgentoptions
...

```

```

download.fileurldestfile
urlhttp://https://file://method = "auto"
method = "auto""internal"file://"libcurl"
"libcurl"capabilities("libcurl")https://curl.se/libcurl/
"libcurl"urldestfile"auto"quiet = FALSE
"internal"file://"wininet"file://http://https://
"wget""curl"method
cacheOK = FALSEhttp://https://available.packages
"libcurl""wget"http://https://"curl"extra = "-L"wgetextra = "--max-redirect=0"
"wininet""libcurl"
urlfile://"internal""wininet""libcurl""curl""wget"
URLencode"wininet"
"wininet""libcurl"
timeoutdownload.fileR_DEFAULT_INTERNET_TIMEOUT

options(timeout = max(300, getOption("timeout")))

```

```

quietinternet.info"libcurl"

```

```

mode = "wb""ab"\n\r\nCRLF
modemissing()url.gz.bz2.xz.tgz.zip.jar.rda.rds.RData.pdfmode = "wb"
mode = "wb""ab"

```

```

0"wget""curl""internal"1url0retvalsurl0
"internal""wininet""libcurl"mode

```

"wininet"

"libcurl""curl"http_proxyftp_proxy<https://curl.se/libcurl/c/libcurl-tutorial.html>

<https://ftps://https://curl.se/docs/sslcerts.html>

method = "libcurl"libcurlSchannellibcurlVersion()"Schannel"CURL_CA_BUNDLE
ca-bundle.crtcurl-ca-bundle.crt/etc/curl-ca-bundle.crtCURL_CA_BUNDLE
<https://curl.se/docs/sslcerts.html><https://raw.githubusercontent.com/bagder/curl-ca-bundle/master/ca-bundle.crt>CURL_CA_BUNDLE

method = "libcurl"libcurlSchannelR_LIBCURL_SSL_REVOKE_BEST_EFFORTTRUE

methodwgetcurlSys.which

download.file

methodlibcurlcapabilities("libcurl")

ftp://

<https://ftps://>

libcurllibcurllibcurlVersion()

"libcurl"

"wget""curl""libcurl""wininet"

"wget"wget

wget<https://www.gnu.org/software/wget/>

curl<https://curl.se/>

optionsHTTPUserAgenttimeoutinternet.info

url

url.showavailable.packagesdownload.packages

download.packages

```
download.packages(pkgs, destdir, available = NULL,
                  repos = getOption("repos"),
                  contriburl = contrib.url(repos, type),
                  method, type = getOption("pkgType"), ...)
```

pkgs

destdir

available [available.packages](#)NULLavailable.packages

repos "https://cran.r-project.org""https://cloud.r-project.org"

contriburl contribrepos

method [download.file](#)

type [install.packages](#)

... [download.file](#)[available.packages](#)

```
download.packagesdestdir"file:""file:""file:///"urlfile://
```

```
download.packagetype = "both"
```

[available.packages](#)[contrib.url](#)

[install.packages](#)

[download.file](#)

edit

```
edit(name, ...)
## Default S3 method:
edit(name = NULL, file = "", title = NULL,
      editor = getOption("editor"), ...)

vi(name = NULL, file = "")
emacs(name = NULL, file = "")
pico(name = NULL, file = "")
xemacs(name = NULL, file = "")
xedit(name = NULL, file = "")

name          namefile
file
title
editor        EDITORVISUALvi"internal"
              editornamefiletitle
...

editeditorname
data.entryeditdata.entry
editnamenameditnamefix
edit(name)editnameeditorfileedit()
dput.deparseOptsdumpedit.data.frame
title

viemacspicoxemacsxedit

edit.data.framedata.entryfix

## Not run:
# use xedit on the function mean and assign the changes
mean <- edit(mean, editor = "xedit")

# use vi on mean and write the result to file mean.out
vi(mean, file = "mean.out")

## End(Not run)
```

edit.data.frame

```
## S3 method for class 'data.frame'
edit(name, factor.mode = c("character", "numeric"),
      edit.row.names = any(row.names(name) != 1:nrow(name)), ...)
```

```
## S3 method for class 'matrix'
edit(name, edit.row.names = !is.null(dn[[1]]), ...)
```

```
name
factor.mode
edit.row.names NULL
...
```

```
is.numeric
NANA
```

```
edit.row.names = FALSEedit.row.names = TRUErow.namesrow223seq(length = nrow)
col7edit.row.names = FALSENULLedit.row.names = TRUErow.namesrow223
```

```
fix(dataframe)
edit
```

[data.entryedit](#)

```
## Not run:
edit(InsectSprays)
edit(InsectSprays, factor.mode = "numeric")

## End(Not run)
```

example

topic\dontrun\dontshow\donttest

```
example(topic, package = NULL, lib.loc = NULL,
  character.only = FALSE, give.lines = FALSE, local = FALSE,
  type = c("console", "html"), echo = TRUE,
  verbose = getOption("verbose"),
  setRNG = FALSE, ask = getOption("example.ask"),
  prompt.prefix = abbreviate(topic, 6),
  catch.aborts = FALSE,
  run.dontrun = FALSE, run.donttest = interactive())
```

topic	help
package	NULL
lib.loc	NULLNULL
character.only	topic
give.lines	
local	TRUEFALSE
type	knitr setRNGlib.loc
echo	TRUE
verbose	TRUE
setRNG	FALSEsetRNG = TRUER CMD check setRNG = {RNGkind("default", "default", "default"); set.seed(1)}
ask	"default" devAskNewPage (ask = TRUE)"default"echo
prompt.prefix	echo
catch.aborts	source ()
run.dontrun	\dontrun
run.donttest	\donttest

lib.loc.[libPaths](#)()lib.loc

local = TRUEexample

\dontrun

\dontshow example()\testonly

\donttest run.donttest = [interactive](#)()example()\donttest

\dontdiff \geq exampleRdiff

give.linescharacter

demo

```
example(InsectSprays)
## force use of the standard package 'stats':
example("smooth", package = "stats", lib.loc = .Library)

## set RNG *before* example as when R CMD check is run:

r1 <- example(quantile, setRNG = TRUE)
x1 <- rnorm(1)
u <- runif(1)
## identical random numbers
r2 <- example(quantile, setRNG = TRUE)
x2 <- rnorm(1)
stopifnot(identical(r1, r2))
## but x1 and x2 differ since the RNG state from before example()
## differs and is restored!
x1; x2

## Exploring examples code:
## How large are the examples of "lm..." functions?
lmex <- sapply(apropos("^lm", mode = "function"),
               example, character.only = TRUE, give.lines = TRUE)
lengths(lmex)
```

file.edit

```
file.edit(..., title = file, editor = getOption("editor"),
          fileEncoding = "")
```

```
...           path.expand
title
editor
fileEncoding  file
```

title

editor"internal"editorEDITORVISUALvi
editornamefiletitle

[filesfile.showeditfix](#)

```
## Not run:  
# open two R scripts for editing  
file.edit("script1.R", "script2.R")  
  
## End(Not run)
```

file_test

file_test(op, x, y)

op x"-f""-d""-L""-h""-x""-w""-r""-nt""-ot"
xy

stat
"-x"[file.access](#)

[file.exists](#)test -e
[file.path](#)[file.info](#)

```
dir <- file.path(R.home(), "library", "stats")  
file_test("-d", dir)  
file_test("-nt", file.path(dir, "R"), file.path(dir, "demo"))
```

`findCRANmirror`

```
findCRANmirror(type = c("src", "web"))
```

```
type
```

```
"https://CRAN.R-project.org"
```

```
  R_CRAN_SRCR_CRAN_WEBtype  
getOption("repos")CRAN"@CRAN@")  
  CRANrepositoriessetRepositories"@CRAN@"  
getOption("repos")R_CRAN_WEB
```

```
setRepositorieschooseCRANmirror
```

```
c(findCRANmirror("src"), findCRANmirror("web"))  
  
Sys.setenv(R_CRAN_WEB = "https://cloud.r-project.org")  
c(findCRANmirror("src"), findCRANmirror("web"))
```

`findLineNum`

```
keep.source = TRUE
```

```
findLineNum(srcfile, line, nameonly = TRUE,  
            envir = parent.frame(), lastenv)
```

```
setBreakpoint(srcfile, line, nameonly = TRUE,  
              envir = parent.frame(), lastenv, verbose = TRUE,  
              tracer, print = FALSE, clear = FALSE, ...)
```

```

srcfile
line
nameonly      TRUEbasename(srcfile)
envir
lastenv
verbose
tracer        tracertracebrowser
print         printtrace
clear         TRUEuntrace
...           trace

```

```

findLineNumenvirlastenv
lastenvenviremptyenv()envir
envirenvironment(envir)
setBreakpointtraceuntracefindLineNum
srcfile"srcfile""filename.R#nn"nnline
wheretraceenvirfindLineNumsetBreakpointlastenvenvir = environment(foo)lastenv =
globalenv()
lastenv = emptyenv()findLineNum

```

```

findLineNumprint
setBreakpointtraceuntrace

```

trace

```

## Not run:
# Find what function was defined in the file mysource.R at line 100:
findLineNum("mysource.R#100")

# Set a breakpoint in both copies of that function, assuming one is in the
# same namespace as myfunction and the other is on the search path
setBreakpoint("mysource.R#100", envir = myfunction)

## End(Not run)

```

fix

fixeditxx

fix(x, ...)

x

... edit

x

editedit.data.frame

editedit.data.frame

```
## Not run:
## Assume 'my.fun' is a user defined function :
fix(my.fun)
## now my.fun is changed
## Also,
fix(my.data.frame) # calls up data editor
fix(my.data.frame, factor.mode="char") # use of ...

## End(Not run)
```

flush.console

flush.console()

format

```
formatUL(x, label = "*", offset = 0,
         width = 0.9 * getOption("width"))
formatOL(x, type = "arabic", offset = 0, start = 1,
         width = 0.9 * getOption("width"))
```

x

label

offset

width

type "arabic""Alph""alph""Roman""roman" type "1""A""a""I""i"

start

[formatDL](#)

```
## A simpler recipe.
x <- c("Mix dry ingredients thoroughly.",
      "Pour in wet ingredients.",
      "Mix for 10 minutes.",
      "Bake for one hour at 300 degrees.")
## Format and output as an unordered list.
writeLines(formatUL(x))
## Format and output as an ordered list.
writeLines(formatOL(x))
## Ordered list using lower case roman numerals.
writeLines(formatOL(x, type = "i"))
## Ordered list using upper case letters and some offset.
writeLines(formatOL(x, type = "A", offset = 5))
```

getAnywhere

getAnywhere()argsAnywhere()

getAnywhere(x)
argsAnywhere(x)

x

getAnywhere()"getAnywhere"

name

objs

where

visible

dups

print[

argsAnywhere()[args](#)

[getS3method](#)getAnywhere

[getgetFromNamespace](#)[args](#)

getAnywhere("format.dist")
getAnywhere("simpleLoess") # not exported from stats
argsAnywhere(format.dist)

`getFromNamespace`

`assignInMyNamespace``getFromNamespace(x, ns, pos = -1, envir = as.environment(pos))``assignInNamespace(x, value, ns, pos = -1,
 envir = as.environment(pos))``assignInMyNamespace(x, value)``fixInNamespace(x, ns, pos = -1, envir = as.environment(pos), ...)``x``value``ns``pos` `get``envir``...` `edit``assignInMyNamespace``ns = "stats"getns<namespace:foo>``getFromNamespace:::``fixInNamespaceeditfixx``getFromNamespace``assignInNamespaceassignInMyNamespacefixInNamespace``assignInNamespace``assignInNamespaceassignInMyNamespacefixInNamespace``getfixgetS3method`

```

getFromNamespace("findGeneric", "utils")
## Not run:
fixInNamespace("predict.ppr", "stats")
stats::predict.ppr
getS3method("predict", "ppr")
## alternatively
fixInNamespace("predict.ppr", pos = 3)
fixInNamespace("predict.ppr", pos = "package:stats")

## End(Not run)

```

getParseData

```
"keep.source"TRUE
```

```

getParseData(x, includeText = NA)
getParseText(parseData, id)

```

```

x                parse
includeText
parseData        getParseData
id

```

```
srcfilegetParseData
```

```

getParseData
NULL

line1          "parse"getSrcLocation#line
col1           "column"getSrcLocation
line2
col2
id
parent         id
token
terminal
text           includeTextTRUENAincludeText == FALSE
id"srcfile"srcfile
getParseText
idparseData

```

getParseData

```
col1
idgetParseData
#line
```

<https://github.com/halpo/parser>

[parsesrcref](#)

```
fn <- function(x) {
  x + 1 # A comment, kept as part of the source
}

d <- getParseData(fn)
if (!is.null(d)) {
  plus <- which(d$token == "'+'")
  sum <- d$parent[plus]
  print(d[as.character(sum),])
  print(getParseText(d, sum))
}
```

getS3method

getS3method(f, class, optional = FALSE, envir = parent.frame())

```
f
class
optional
envir      environment
```

[get](#)

fgetS3methodf

NULLoptional = TRUE

[methodsgetgetAnywhere](#)

```
require(stats)
exists("predict.ppr") # false
getS3method("predict", "ppr")
```

getWindowHandle

getWindowHandle(which = "Console")

which [windows](#)

getWindowHandlewhich

"Console"
"Frame"
"Process"

NULLwhich

NULL

[getIdentificationgetWindowHandles](#)

```
if(.Platform$OS.type == "windows")
  print( getWindowHandle() )
```

`getWindowsHandles`

Rgui

```
getWindowsHandles(which = "R", pattern = "", minimized = FALSE)
```

which

pattern

minimized

[arrangeWindows](#)

```
which"R"Rgui
```

```
"all"
```

```
patterngrep
```

```
minimized = FALSE
```

[arrangeWindowsgetWindowsHandle](#)

```
if(.Platform$OS.type == "windows") withAutoprint({  
  getWindowsHandles()  
  getWindowsHandles("all")  
})
```

glob2rx

[regexp](#)

[sub\(\)](#)

```
glob2rx(pattern, trim.head = FALSE, trim.tail = TRUE)
```

pattern

trim.head "^.*"

trim.tail ".*\$"

?.*.*

([{ patternglob2rx() pattern

pattern

[regexpsubSys.glob](#)

```
stopifnot(glob2rx("abc.*") == "^abc\\.","
  glob2rx("a?b.*") == "^a.b\\.","
  glob2rx("a?b.*", trim.tail = FALSE) == "^a.b\\..*$",
  glob2rx("*.doc") == "^.*\\.doc$",
  glob2rx("*.doc", trim.head = TRUE) == "\\\\.doc$",
  glob2rx("*.t*") == "^.*\\.t",
  glob2rx("*.t??") == "^.*\\.t..$",
  glob2rx("[*]") == "^.*\\["
)
```

globalVariables

```
globalVariablescheckglobalVariables
setRefClass()
suppressForeignCheck.NAMEcheckFF(registration = TRUE).Call.ExternaldontCheck
```

```
globalVariables(names, package, add = TRUE)
suppressForeignCheck(names, package, add = TRUE)
```

```
names
package
                                globalVariablessuppressForeignCheck
add                                names
```

```
globalVariablessuppressForeignCheck
```

```
globalVariablessuppressForeignCheck
```

```
globalVariables
suppressForeignCheck
```

```
check
```

dontCheck

```
## Not run:
## assume your package has some code that assigns ".obj1" and ".obj2"
## but not in a way that codetools can find.
## In the same source file (to remind you that you did it) add:
if(getRversion() >= "2.15.1") utils::globalVariables(c(".obj1", ".obj2"))

## To suppress messages about a run-time calculated native symbol,
## save it to a local variable.

## At top level, put this:
if(getRversion() >= "3.1.0") utils::suppressForeignCheck("localvariable")
```

```
## Within your function, do the call like this:
localvariable <- if (condition) entry1 else entry2
.Call(localvariable, 1, 2, 3)

## HOWEVER, it is much better practice to write code
## that can be checked thoroughly, e.g.
if(condition) .Call(entry1, 1, 2, 3) else .Call(entry2, 1, 2, 3)

## End(Not run)
```

hashtab

```
hashtab(type = c("identical", "address"), size)
gethash(h, key, nomatch = NULL)
sethash(h, key, value)
remhash(h, key)
numhash(h)
typhash(h)
maphash(h, FUN)
clrhash(h)
is.hashtab(x)
## S3 method for class 'hashtab'
h[[key, nomatch = NULL, ...]]
## S3 replacement method for class 'hashtab'
h[[key, ...]] <- value
## S3 method for class 'hashtab'
print(x, ...)
## S3 method for class 'hashtab'
format(x, ...)
## S3 method for class 'hashtab'
length(x)
## S3 method for class 'hashtab'
str(object, ...)
```

type	character
size	
hobject	
key	
nomatch	key
value	key
FUN	function
x	
...	

environment

```
hashtabidentical()type = "identical" type = "address" "identical" size
gethashkeykeynomatch
sethashremhashkey
maphashFUNFUN
clrhash
```

```
hashtabtype
gethashkeynomatch
sethashvalue
remhashTRUEkeyFALSE
numhash
typhash"identical""address"
maphashclrhashNULL
```

"identical"duplicatedunique

```
identical()ignore.environment = FALSEidentical()
identical()extptr.as.ref = TRUE
```

[[length

```
## Create a new empty hash table.
h1 <- hashtab()
h1
```

```
## Add some key/value pairs.
sethash(h1, NULL, 1)
sethash(h1, .GlobalEnv, 2)
for (i in seq_along(LETTERS)) sethash(h1, LETTERS[i], i)
```

```
## Look up values for some keys.
gethash(h1, NULL)
gethash(h1, .GlobalEnv)
gethash(h1, "Q")
```

```
## Remove an entry.
(remhash(h1, NULL))
gethash(h1, NULL)
(remhash(h1, "XYZ"))
```

```
## Using the element operator.
h1[["ABC"]]
h1[["ABC", nomatch = 77]]
```

```

h1[["ABC"]] <- "DEF"
h1[["ABC"]]

## Integers and real numbers that are equal are considered different
## (not identical) as keys:
identical(3, 3L)
sethash(h1, 3L, "DEF")
gethash(h1, 3L)
gethash(h1, 3)

## Two variables can refer to the same hash table.
h2 <- h1
identical(h1, h2)
## set in one, see in the "other" <==> really one object with 2 names
sethash(h2, NULL, 77)
gethash(h1, NULL)
str(h1)

## An example of using maphash(): get all hashkeys of a hash table:
hashkeys <- function(h) {
  val <- vector("list", numhash(h))
  idx <- 0
  maphash(h, function(k, v) { idx <- idx + 1
                              val[idx] <- list(k) })
  val
}

kList <- hashkeys(h1)
str(kList) # the *order* is "arbitrary" & cannot be "known"

```

hasName

hasName

hasName(x, name)

x

name

hasName(x, name)name %in% names(x)!is.null(x\$name)

nameTRUEnames(x)

[%in%exists](#)

```

x <- list(abc = 1, def = 2)
!is.null(x$abc) # correct
!is.null(x$a)   # this is the wrong test!
hasName(x, "abc")
hasName(x, "a")

```

head

```
head()tail()"ts"
```

```

head(x, ...)
## Default S3 method:
head(x, n = 6L, ...)
## S3 method for class 'matrix'
head(x, n = 6L, ...) # is exported as head.matrix()
## NB: The methods for 'data.frame' and 'array' are identical to the 'matrix' one

```

```

## S3 method for class 'ftable'
head(x, n = 6L, ...)
## S3 method for class 'function'
head(x, n = 6L, ...)

```

```

tail(x, ...)
## Default S3 method:
tail(x, n = 6L, keepnums = FALSE, addrownums, ...)
## S3 method for class 'matrix'
tail(x, n = 6L, keepnums = TRUE, addrownums, ...) # exported as tail.matrix()
## NB: The methods for 'data.frame', 'array', and 'table'
##      are identical to the 'matrix' one

```

```

## S3 method for class 'ftable'
tail(x, n = 6L, keepnums = FALSE, addrownums, ...)
## S3 method for class 'function'
tail(x, n = 6L, ...)

```

```
.checkHT(n, d)
```

```

x
n          dim(x)logicaln[i]n[i]abs(n[i])NAlength(n) < length(dim(x))
keepnums   dim(x)NULL
addrownums  keepnumskeepnumskeepnums
...
d          dim(x)NULL

```

```
head()tail()xhead.matrix()tail.matrix()
head()tail()dim()length()dim()NULL[drop
```

```
xtail()xkn[k]dimnames(x)[[k]]dimnames(x)NULL
```

```
k=1 "[n,]"
```

```
k=2 "[,n]"
```

```
k>2 "n"
```

```
keepnums = FALSE
```

```
data.frameattributeshead()tail()
```

```
.checkHT(d, n)head(x, n)tail(x, n)d <- dim(x)n
```

```
xarrayx[., drop=FALSE]ftablexformat(x)
```

```
tailkeepnumsTRUE>2arrtail(arr, c(2,2,-1))[ , , 2]tail(arr, c(2,2,-1))[ , , "2"]
```

```
head(letters)
head(letters, n = -6L)
```

```
head(freeny.x, n = 10L)
head(freeny.y)
```

```
head(gait) # 3d array
head(gait, c(6L, 2L))
head(gait, c(6L, 2L, -1L))
```

```
tail(letters)
tail(letters, n = -6L)
```

```
tail(freeny.x)
## the bottom-right "corner" :
tail(freeny.x, n = c(4, 2))
tail(freeny.y)
```

```
tail(gait)
tail(gait, c(6L, 2L))
tail(gait, c(6L, 2L, -1L))
```

```
## gait without dimnames --> keepnums showing original row/col numbers
a3 <- gait ; dimnames(a3) <- NULL
tail(a3, c(6, 2, -1))# keepnums = TRUE is default here!
tail(a3, c(6, 2, -1), keepnums = FALSE)
```

```
## data frame w/ a (non-standard) attribute:
```

```

treeS <- structure(trees, foo = "bar")
(n <- nrow(treeS))
stopifnot(exprs = { # attribute is kept
  identical(htS <- head(treeS), treeS[1:6, ])
  identical(attr(htS, "foo") , "bar")
  identical(tlS <- tail(treeS), treeS[(n-5):n, ])
  ## BUT if I use "useAttrib(.)", this is *not* ok, when n is of length 2:
  ## --- because [i,j]-indexing of data frames *also* drops "other" attributes ..
  identical(tail(treeS, 3:2), treeS[(n-2):n, 2:3] )
})

tail(library) # last lines of function

head(stats::ftable(Titanic))

## 1d-array (with named dim) :
a1 <- array(1:7, 7); names(dim(a1)) <- "02"
stopifnot(exprs = {
  identical( tail(a1, 10), a1)
  identical( head(a1, 10), a1)
  identical( head(a1, 1), a1 [1 , drop=FALSE] ) # was a1[1] in R <= 3.6.x
  identical( tail(a1, 2), a1[6:7])
  identical( tail(a1, 1), a1 [7 , drop=FALSE] ) # was a1[7] in R <= 3.6.x
})

```

help

help

```

help(topic, package = NULL, lib.loc = NULL,
      verbose = getOption("verbose"),
      try.all.packages = getOption("help.try.all.packages"),
      help_type = getOption("help_type"))

```

topic

topic
topic

package NULL(pkg_ref)

lib.loc NULLNULL

verbose TRUE

try.all.packages

Note

help_type "text" "html" "pdf"

[browseURL](#)

[startDynamicHelp](#)

[help](#)

[options](#)(useFancyQuotes = FALSE)

[topic](#)

[lib.loc](#)

[help](#)

[function](#)[ifelse](#)[for](#)[in](#)[repeat](#)[while](#)[break](#)[next](#)[reserved](#)[TRUE](#)[NA](#)[Inf](#)

[topic](#)[getOption](#)("menu.graphics")

[lib.loc](#)[libPaths](#)()

[pdf](#)[latex](#)

[R](#)[help.cfg](#)[Rd.sty](#)[R_PAPERSIZE](#)[getOption](#)("papersize")[R_RD4PDF](#)

[offline_help](#)[help](#)[pertex](#)[inputs](#)

[lib.loc](#)[lib.loc](#)

[try.all.packages](#)[TRUE](#)[packages](#)[lib.loc](#)[topic](#)[help_type](#) = "html"

[?](#)

[help.search](#)()[??](#)[help.start](#)()[library](#)()[data](#)()[methods](#)()

[prompt](#)()[help](#)

```

help()
help(help)          # the same

help(lapply)

help("for")          # or ?"for", but quotes/backticks are needed

try({# requires working TeX installation:
  help(dgamma, help_type = "pdf")
  ## -> nicely formatted pdf -- including math formula -- for help(dgamma):
  system2(getOption("pdfviewer"), "dgamma.pdf", wait = FALSE)
})

help(package = "splines") # get help even when package is not loaded

topi <- "women"
help(topi)

try(help("bs", try.all.packages = FALSE)) # reports not found (an error)
help("bs", try.all.packages = TRUE)       # reports can be found
                                           # in package 'splines'

## For programmatic use:
topic <- "family"; pkg_ref <- "stats"
help((topic), (pkg_ref))

```

help.request

```

help.request(subject = "",
             address = "r-help@R-project.org",
             file = "R.help.request", ...)

```

```

subject      '
address
file
...          methodccaddresscreate.post

```

<https://www.r-project.org/posting-guide.html>

help.request

[create.post](#)

`bug.report()`

`https://www.r-project.org/posting-guide.htmlsessionInfo()
create.post`

`help.search`

```
help.search(pattern, fields = c("alias", "concept", "title"),
             apropos, keyword, whatis, ignore.case = TRUE,
             package = NULL, lib.loc = NULL,
             help.db = getOption("help.db"),
             verbose = getOption("verbose"),
             rebuild = FALSE, agrep = NULL, use_UTF8 = FALSE,
             types = getOption("help.search.types"))
??pattern
field??pattern
```

pattern	aproposkeywordwhatis
fields	"name""title""alias""concept""keyword"
apropos	
keyword	R.home("doc")/KEYWORDSkeyworddagrepFALSE
whatis	
ignore.case	TRUEFALSE
package	NULLlib.loc
lib.loc	NULLNULL
help.db	NULL
verbose	TRUETRUE21verbose = 1verbose >= 2
rebuild	lib.locpackage
agrep	NULLkeywordagrepgrepFALSEmax.distanceagrep
use_UTF8	agrep
types	"vignette""help""demo"
field	fields


```
hsearch.rdsMetaMeta/vignette.rds
```

```
aproposwhat is
```

```
agrep = FALSE
```

```
???::::
```

```
\keywordKEYWORDSinternalKEYWORDS\concept
```

```
"name""alias""concept"\VignetteKeyword"keyword""concept"
```

```
"hsearch"
```

```
R.app
```

```
helphelp.start
```

```
RSiteSearch
```

```
apropos
```

```
## Not run:
```

```
help.search("linear models") # In case you forgot how to fit linear models
```

```
## End(Not run)
```

```
help.search("non-existent topic")
```

```
??utils::help # All the topics matching "help" in the utils package
```

```
## Documentation with topic/concept/title matching 'print'
```

```
## (disabling fuzzy matching to not also match 'point')
```

```
help.search("print", agrep = FALSE)
```

```
help.search(apropos = "print", agrep = FALSE) # ignores concepts
```

```
## Help pages with documented topics starting with 'try':
```

```
help.search("^try", fields = "alias")
```

```
alias??"^try" # the same
```

```
## Help pages documenting high-level plots:
```

```
help.search(keyword = "hplot")
```

```
RShowDoc("KEYWORDS") # show all keywords
```

help.start

```
help.start(update = FALSE, gui = "irrelevant",  
           browser = getOption("browser"), remote = NULL)
```

```
update          remoteNULL  
gui  
browser         PATH  
remote
```

```
remotestartDynamicHelp  
/html/packages.htmlmake.packages.html().libPathsupdate = TRUE  
remote
```

```
help()  
browseURL  
RSiteSearch
```

```
help.start()  
  
## the 'remote' arg can be tested by  
help.start(remote = paste0("file://", R.home()))
```

hsearch-utils

```
hsearch_db(package = NULL, lib.loc = NULL,  
            types = getOption("help.search.types"),  
            verbose = getOption("verbose"),  
            rebuild = FALSE, use_UTF8 = FALSE)  
hsearch_db_concepts(db = hsearch_db())  
hsearch_db_keywords(db = hsearch_db())
```

package	NULLlib.loc
lib.loc	NULLNULL
types	help.search
verbose	help.search
rebuild	help.search
use_UTF8	
db	hsearch_db()

hsearch_db()[help.search](#)rebuild = TRUE

[help.search](#)

hsearch_db_concepts()hsearch_db_keywords()

```
db <- hsearch_db()
## Total numbers of documentation objects, aliases, keywords and
## concepts (using the current format):
sapply(db, NROW)
## Can also be obtained from print method:
db
## 10 most frequent concepts:
head(hsearch_db_concepts(), 10)
## 10 most frequent keywords:
head(hsearch_db_keywords(), 10)
```

INSTALL

R CMD INSTALL [options] [-l lib] pkgs

pkgs

lib --library=lib

options R CMD INSTALL --help

```
pkgs
R CMD INSTALL pkgslib
R CMD INSTALL -l lib
libpkgspkgsgzipbzip2xzcompressR CMD INSTALL --build
untarR_INSTALL_TARR_INSTALL_TARtar.exe
--preclean--clean
configure--configure-args--configure-varsLIBSCPPFLAGS--configure-vars
--no-configure
```

```
install.packages
--buildutils::tarR_INSTALL_TAR
--html--no-html
keep.sourcesource--with-keep.sourceR_KEEP_PKG_SOURCEyes
--install-teststestsR_ALWAYS_INSTALL_TESTS--install-tests
R CMD INSTALL --help
```

```
src/Makefilesrc/Makefile.winR CMD INSTALL
R CMD INSTALL--no-multiarch--libs-only
R CMD INSTALL --merge-multiarch mypkg_version.tar.gz
```

```
StagedInstall: noDESCRIPTION--no-staged-installR_INSTALL_STAGEDfalseno
--pkglock--lock
```

```
pkgs
INSTALLtempdir/tmp/tmpnoexecTMPDIR
untarzsdtarzd4
```

```
REMOVE.libPathsinstall.packagesupdate.packages
RShowDocdoc/manual
```

install.packages

```
install.packages(pkgs, lib, repos = getOption("repos"),
  contriburl = contrib.url(repos, type),
  method, available = NULL, destdir = NULL,
  dependencies = NA, type = getOption("pkgType"),
  configure.args = getOption("configure.args"),
  configure.vars = getOption("configure.vars"),
  clean = FALSE, Ncpus = getOption("Ncpus", 1L),
  verbose = getOption("verbose"),
  libs_only = FALSE, INSTALL_opts, quiet = FALSE,
  keep_outputs = FALSE, ...)
```

pkgs

```
repos = NULL
.ziphttp://file://type = "source"
R CMD build --binaryhttp://file://.tgz
```

lib [.libPaths\(\)](#)

repos ["https://cloud.r-project.org"](https://cloud.r-project.org)[url](#)
NULLpkgs

contriburl contribrepostype = "both"

method [download.file](#)

available [available.packages](#)NULLavailable.packagestype = "both"

destdir NULLdownloaded_packages

dependencies repos = NULLc("Depends", "Imports", "LinkingTo", "Suggests",
"Enhances")

```
lib
NAc("Depends", "Imports", "LinkingTo")
TRUEc("Depends", "Imports", "LinkingTo", "Suggests")pkgs
c("Depends", "Imports", "LinkingTo")pkgs
"LinkingTo"
```

type "source"

configure.args --configure-argsR CMD INSTALL--configure-args
--configure-args

configure.vars configure.args--configure-argsconfigure

clean --cleanR CMD INSTALL

Ncpus makeSys.getenv("MAKE", "make")-k -j <Ncpus>

verbose

```

libs_only      --libs-onlyINSTALL_opts
INSTALL_opts   R CMD INSTALLc("--html", "--no-multiarch", "--no-test-load")
               "--dsym"

```

```

quiet          available.packages()
keep_outputs   .out
...            download.fileavailable.packages"lock"

```

```

.libPaths()libSys.getenv("R_LIBS_USER")

```

```

libR_LIBS
update.packagesinstall.packages

```

```

NULL

```

```

type"binary""win""mac""binary""big-sur-arm64""win.binary"
"mac.binary.big-sur-arm64".Platform$pkgType"source"

```

```

options(install.packages.check.source = "no")

```

```

"both"getOption("install.packages.compile.from.source")type = "both""binary"
contriburlavailable
type = "source"
PATH
install.packages

```

```

00LOCK--pkglockNcpus > 1L--no-lock
lockTRUEgetOption("install.lock", FALSE)lockgetOption("install.lock", TRUE)
"pkglock"
libs_only = TRUE
--pkglock

```

```

pkgsNcpus > 1makemakemake -j dmakepmakemakeMAKEmake
install.packagespkgsavailablepkgsavailable

```

```
R CMD INSTALL_R_INSTALL_PACKAGES_ELAPSED_TIMEOUT_mhs0
timeoutsystem2timeouttimeoutR_TIMEOUTc:/Windows/system32/timeout.exeError 124
make
```

package 'RODBC' is not available (for R version 3.5.3)

```
https://cran.r-project.org/package=RODBCDependsOld sourcesinstall.packages()
available.packages(filters = "OS_type")[, ]
```

```
INSTALLR-develinstall.packagetype = "source"
```

```
install.packages.dll
```

```
update.packagesavailable.packagesdownload.packagesinstalled.packagescontrib.url
download.file
untar
INSTALLREMOVEremove.packageslibrary.packagesread.dcf
```

```
## Not run:
## A Linux example for Fedora's layout of udunits2 headers.
install.packages(c("ncdf4", "RNetCDF"),
  configure.args = c(RNetCDF = "--with-netcdf-include=/usr/include/udunits2"))

## End(Not run)
```

```
installed.packages
```

```
installed.packages(lib.loc = NULL, priority = NULL,
  noCache = FALSE,
  cache_user_dir =
    str2logical(Sys.getenv("R_PACKAGES_CACHE_USER_DIR",
      FALSE)),
  fields = NULL,
  subarch = .Platform$r_arch, ...)
```

```

lib.loc      NULL.libPaths
priority     NULL"high"c("base", "recommended")priority = NA_character_
noCache
cache_user_dir logicalR_user_dir("base", "cache")tempdir()
fields       DESCRIPTIONNULLNA
subarch      NULL
...

installed.packagesDESCRIPTIONlib.loc
fieldsnoCache = TRUE

"Package""LibPath""Version""Priority""Depends""Imports""LinkingTo""Suggests"
"Enhances""OS_type""License""Built"fields

```

```

find.packagesystem.filerequireNamespacerequirepackageDescription

```

```

update.packagesinstall.packagesINSTALLREMOVE

```

```

## confine search to .Library for speed
str(ip <- installed.packages(.Library, priority = "high"))
ip[, c(1,3:5)]
plic <- installed.packages(.Library, priority = "high", fields = "License")
## what licenses are there:
table( plic[, "License"] )

## Recommended setup (by many pros):
## Keep packages that come with R (priority="high") and all others separate!
## Consequently, .Library, R's "system" library, shouldn't have any
## non-"high"-priority packages :
pSys <- installed.packages(.Library, priority = NA_character_)
length(pSys) == 0 # TRUE under such a setup

```

```

isS3method

```

```

methodfclassfpaste(f, class, sep=".")

```

```

isS3method(method, f, class, envir = parent.frame())

```



```
method      "."fclass
f           method
class       method
envir       environmentgetS3method()
```

```
logicalTRUEFALSE
```

```
methodsgetS3method
```

```
isS3method("t")          # FALSE - it is an S3 generic
isS3method("t.default")  # TRUE
isS3method("t.ts")       # TRUE
isS3method("t.test")     # FALSE
isS3method("t.data.frame")# TRUE
isS3method("t.lm")       # FALSE - not existing
isS3method("t.foo.bar")  # FALSE - not existing

## S3 methods with "4 parts" in their name:
ff <- c("as.list", "as.matrix", "is.na", "row.names", "row.names<-")
for(m in ff) if(isS3method(m)) stop("wrongly declared an S3 method: ", m)
(m4 <- paste(ff, "data.frame", sep="."))
for(m in m4) if(!isS3method(m)) stop("not an S3 method: ", m)
```

```
isS3stdGeneric
```

```
f
```

```
isS3stdGeneric(f)
```

```
f
```

```
UseMethodUseMethod
```

```
f"traceable"
```

```
fTRUEUseMethodFALSE
```

LINK

R CMD LINK [options] linkcmd

linkcmd

options

libtool/bin

R CMD LINK --help

LINKR-devel

COMPILE

```
## Not run: ## examples of front-ends linked against R.
## First a C program
CC=`R CMD config CC`
R CMD LINK $CC -o foo foo.o `R CMD config --ldflags`

## if Fortran code has been compiled into ForFoo.o
FLIBS=`R CMD config FLIBS`
R CMD LINK $CC -o foo foo.o ForFoo.o `R CMD config --ldflags` $FLIBS

## And for a C++ front-end
CXX=`R CMD config CXX`
R CMD COMPILE foo.cc
R CMD LINK $CXX -o foo foo.o `R CMD config --ldflags`

## End(Not run)
```

localeToCharset

```
localeToCharset(locale = Sys.getlocale("LC_CTYPE"))
```

```
locale
```

```
"English_United Kingdom.1252"  
es_MX.iso88591utf8es  
C"ASCII"
```

```
NA
```

```
libiconvlibc
```

```
Sys.getlocaleiconv
```

```
localeToCharset()
```

ls.str

```
ls.strlsf.strlsstr()
```

```
ls.str(pos = -1, name, envir, all.names = FALSE,  
       pattern, mode = "any")
```

```
lsf.str(pos = -1, envir, ...)
```

```
## S3 method for class 'ls_str'  
print(x, max.level = 1, give.attr = FALSE, ...,  
      digits = max(1, getOption("str")$digits.d))
```

```

pos          search-1
name         searchls
envir        ls
all.names    .ls
pattern      lspattern
max.level
give.attr    TRUE
mode         modeexistsget
x            "ls_str"
...          lsf.strls.strlsprint.ls_strstr
digits

```

```
ls.strlsf.str"ls_str"lsf.strlsprint()str()
```

[strsummaryargs](#)

```

require(stats)

lsf.str() #- how do the functions look like which I am using?
ls.str(mode = "list") #- what are the structured objects I have defined?

## create a few objects
example(glm, echo = FALSE)
ll <- as.list(LETTERS)
print(ls.str(), max.level = 0)# don't show details

## which base functions have "file" in their name ?
lsf.str(pos = length(search()), pattern = "file")

## demonstrating that ls.str() works inside functions
## ["browser/debug mode"]:
tt <- function(x, y = 1) { aa <- 7; r <- x + y; ls.str() }
(nms <- sapply(strsplit(capture.output(tt(2))," *:"), `[, 1)`)
stopifnot(setequal(nms, c("aa", "r", "x", "y")))

```

`maintainer`

`maintainer(pkg)`

`pkg`

[bug.report](#)

`NA_character_`

`<d.scott@auckland.ac.nz><source@sharpsteen.net>`

<https://stat.ethz.ch/pipermail/r-help/2010-February/230027.html>

[packageDescriptionbug.report](#)

`maintainer("MASS")`

`make.packages.html`

```
make.packages.html(lib.loc = .libPaths(), temp = FALSE,  
                   verbose = TRUE, docdir = R.home("doc"))
```

`lib.loc`

`temp`

`verbose`

`docdir` `temphtmlpackages.html`

```
packages.htmlhelp.startR.home("doc")/html
```

```
DESCRIPTION
```

```
temp = TRUElib.loc
```

```
FALSE
```

```
help.start
```

```
## Not run:
```

```
make.packages.html()
```

```
# this can be slow for large numbers of installed packages.
```

```
## End(Not run)
```

```
make.socket
```

```
server = FALSEserver = TRUEon.exit
```

```
make.socket(host = "localhost", port, fail = TRUE, server = FALSE)
```

```
host
```

```
port
```

```
fail
```

```
server
```

```
"socket"
```

```
socket
```

```
port
```

```
host
```

```
server = TRUE
```

```
XLISP-Stat
```

```
close.socketread.socket
capabilities("sockets")
```

```
daytime <- function(host = "localhost"){
  a <- make.socket(host, 13)
  on.exit(close.socket(a))
  read.socket(a)
}
## Official time (UTC) from US Naval Observatory
## Not run: daytime("tick.usno.navy.mil")
```

menu

menu0

```
menu(choices, graphics = FALSE, title = NULL)
```

```
choices
graphics
title          NULL
```

```
graphics = TRUEmenu
```

```
titleNULL""
```

```
select.list
```

```
## Not run:
switch(menu(c("List letters", "List LETTERS"))) + 1,
      cat("Nothing done\n"), letters, LETTERS)

## End(Not run)
```

methods

```
methods(generic.function, class, all.names = FALSE, dropPath = FALSE)
.S3methods(generic.function, class, envir = parent.frame(),
            all.names = FALSE, dropPath = FALSE, useEnv = FALSE)
```

```
## S3 method for class 'MethodsFunction'
format(x, byclass = attr(x, "byclass"), ...)
## S3 method for class 'MethodsFunction'
print(x, byclass = attr(x, "byclass"), ...)
```

generic.function

```
class          generic.function
envir
all.names      logicalFALSE.
dropPath       logicalsearch().GlobalEnvpackage:basebaseenv()FALSEprint()
               dropPath=TRUEsearch()
useEnv         logicalenvirdropPath=TRUEsearch()
x              methods(.)"MethodsFunction"
byclass        logical"byclass"
...
```

```
methods()generic.functionclassUseMethodsearch().S3methods().S4methods()
generic.function"byclass"FALSEprintgeneric.class*generic.class-method
generic,A,B-method
class"byclass"TRUEprintgeneric
generic.classgetAnywheregetS3methodgetMethod
??"generic<tab>"
```

```
"MethodsFunction""byclass""info""byclass"logicalclass"info"
```

character

attach()search()

factor

methods

[S3MethodsclassgetS3method](#)
[getMethodshowMethodsMethods_Details](#)

```
methods(class = "MethodsFunction") # format and print

require(stats)

methods(summary)
methods(class = "aov")      # S3 class
## The same, with more details and more difficult to read:
print(methods(class = "aov"), byclass=FALSE)
methods("[[")               # uses C-internal dispatching
methods("$")
methods("$<-")               # replacement function
methods("+")                # binary operator
methods("Math")             # group generic
require(graphics)
methods(axis)               # looks like a generic, but is not

mf <- methods(format)       # quite a few; ... the last few :
tail(cbind(meth = format(mf)))

if(require(Matrix, quietly = TRUE)) {
  print(methods(class = "Matrix")) # S4 class
  m <- methods(dim)             # S3 and S4 methods
  print(m)
  print(attr(m, "info"))        # more extensive information

  ## --> help(showMethods) for related examples
}
```

mirrorAdmin

```
mirror2html(mirrors = NULL, file = "mirrors.html",
  head = "mirrors-head.html", foot = "mirrors-foot.html")
checkCRAN(method)
```

```
mirrors
file
head
foot
method      download.file
```

```
mirror2html
checkCRAN
```

```
modifyList
```

```
modifyList(x, val, keep.null = FALSE)
```

```
x          list
val        x
keep.null  TRUE NULL val NULL xx
```

```
xvalxxxkeep.nullvalx[[name]]modifyList(x[[name]], val[[name]])
```

```
<Deepayan.Sarkar@R-project.org>
```

```
foo <- list(a = 1, b = list(c = "a", d = FALSE))
bar <- modifyList(foo, list(e = 2, b = list(d = TRUE)))
str(foo)
str(bar)
```

news

```
news(query, package = "R", lib.loc = NULL, format = NULL,
      reader = NULL, db = NULL)
```

```
## S3 method for class 'news_db'
print(x, doBrowse = interactive(),
      browser = getOption("browser"), ...)
```

```
query
package      "R" "R-3" "R-2"
lib.loc      NULL NULL
format
reader
dbx          news()
doBrowse     browseURLhelp.start
browser      browseURL
...          print()
```

```
package "R" NEWS R.home("doc") "R-3" "R-2" inst/NEWS.Rd NEWS.md NEWSinst/NEWS
query NULL queryVersion Category Date Text TRUE Version Date numeric_version Date
```

```
"news_db" Version Category Date Text HTML NA attributes "package" "subset" query
```

```
inst/NEWS.Rd inst/NEWS.Rd\itemize\section strptime
  \section{Changes in version 2.0 (2020-02-02, <note>)}{
    \itemize{
      \item ....
    }
  }
```

```
\subsection NEWS.Rd\encoding
```

```
NEWS.md NEWS.md https://commonmark.org/
```

```
NEWS NEWS
```

```
o*--+
```

```
news2Rd NEWS NEWStools::news2Rd(dir, "NEWS.Rd") codify = TRUE dir NEWS.Rd inst
```

```

## Build a db of all R news entries.
db <- news()

## Bug fixes with PR number in 4.0.0.
db4 <- news(Version == "4.0.0" & grepl("^BUG", Category) & grepl("PR#", Text),
            db = db)
nrow(db4)

## print db4 to show in an HTML browser.

## News from a date range ('Matrix' is there in a regular R installation):
if(length(iM <- find.package("Matrix", quiet = TRUE)) && nzchar(iM)) {
  dM <- news(package="Matrix")
  stopifnot(identical(dM, news(db=dM)))
  dM2014 <- news("2014-01-01" <= Date & Date <= "2014-12-31", db = dM)
  stopifnot(paste0("1.1-", 2:4) %in% dM2014[, "Version"])
}

## Which categories have been in use? % R-core maybe should standardize a bit more
sort(table(db[, "Category"]), decreasing = TRUE)
## Entries with version >= 4.0.0
table(news(Version >= "4.0.0", db = db)$Version)

## do the same for R 3.x.y, more slowly
db3 <- news(package = "R-3")
sort(table(db3[, "Category"]), decreasing = TRUE)
## Entries with version >= 3.6.0
table(news(Version >= "3.6.0", db = db3)$Version)

```

```

nsl

```

```

gethostbyname

```

```

nsl(hostname)

```

```

hostname

```

```

NULLarpa/inet.h

```

```

NULL

```

```

if(.Platform$OS.type == "unix") # includes Mac
  print( nsl("www.r-project.org") )

```

`object.size`

```
object.size(x)
```

```
## S3 method for class 'object_size'
format(x, units = "b", standard = "auto", digits = 1L, ...)
## S3 method for class 'object_size'
print(x, quote = FALSE, units = "b", standard = "auto",
      digits = 1L, ...)
```

```
x
```

```
quote
```

```
units
```

```
standard
```

```
standard = "legacy" "b" "Kb" "Mb" "Gb" "Tb" "Pb" "B" "KB" "MB" "GB" "TB"
          "PB"
```

```
standard = "IEC" "B" "KiB" "MiB" "GiB" "TiB" "PiB" "EiB" "ZiB" "YiB"
```

```
standard = "SI" "B" "kB" "MB" "GB" "TB" "PB" "EB" "ZB" "YB" "RB" "QB"
```

```
units = "auto" standard = "auto"
```

```
standard
```

```
"legacy" "IEC" "SI" "auto"
```

```
digits
```

```
round
```

```
...
```

```
EXTPTRSXP
```

```
"object_size"
```

```
formatprintstandardunitsunits = "auto"digitsstandard = "auto"units
```

standard

https://en.wikipedia.org/wiki/Binary_prefix

Memory-limits

```
object.size(letters)
object.size(ls)
format(object.size(library), units = "auto")

sl <- object.size(rep(letters, 1000))

print(sl)                                ## 209288 bytes
print(sl, units = "auto")                 ## 204.4 Kb
print(sl, units = "auto", standard = "IEC") ## 204.4 KiB
print(sl, units = "auto", standard = "SI")  ## 209.3 kB

(fsl <- sapply(c("Kb", "KB", "KiB"),
               function(u) format(sl, units = u)))
stopifnot(identical( ## assert that all three are the same :
               unique(substr(as.vector(fsl), 1,5)),
               format(round(as.vector(sl)/1024, 1))))
```

```
## find the 10 largest objects in the base package
z <- sapply(ls("package:base"), function(x)
  object.size(get(x, envir = baseenv()))))
if(interactive()) {
  as.matrix(rev(sort(z))[1:10])
} else # (more constant over time):
  names(rev(sort(z))[1:10])
```

package.skeleton

package.skeletonRead-and-delete-me

```
package.skeleton(name = "anRpackage", list,
  environment = .GlobalEnv,
  path = ".", force = FALSE,
  code_files = character(), encoding = "unknown")
```

```
name
list          listenvironmentcode_files
environment
path
force         FALSE
code_files
encoding      characterEncoding:DESCRIPTION"latin1""latin2""UTF-8"
```

```
listenvironmentcode_filescode_fileslistenvironment.
promptpromptClasspromptMethodspromptImportpackage.skeleton
namepathcode_filescode_files".R"
_zamake.unique(sep = "_")z
R/-internal.R
Read-and-delete-me
```

[INSTALLinstall.packages](#)

[prompt](#)[promptClass](#)[promptMethods](#)
[package_native_routine_registration_skeleton](#)

```

require(stats)
## two functions and two "data sets" :
f <- function(x, y) x+y
g <- function(x, y) x-y
d <- data.frame(a = 1, b = 2)
e <- rnorm(1000)

package.skeleton(list = c("f", "g", "d", "e"), name = "mypkg")

```

packageDescription

DESCRIPTION"packageDescription"

```

packageDescription(pkg, lib.loc = NULL, fields = NULL,
                   drop = TRUE, encoding = "")
packageVersion(pkg, lib.loc = NULL)
packageDate(pkg, lib.loc = NULL,
            date.fields = c("Date", "Packaged", "Date/Publication", "Built"),
            tryFormats = c("%Y-%m-%d", "%Y/%m/%d", "%D", "%m/%d/%y"),
            desc = packageDescription(pkg, lib.loc=lib.loc, fields=date.fields))
asDateBuilt(built)

```

```

pkg
lib.loc      NULLNULL
fields
drop         TRUEfields"packageDescription"
encoding     EncodingNAiconv""
date.fields  as.Date(.)NA
tryFormats   as.Date.character()
desc         listdate.fieldspackageDescription()
built        asDateBuilt()characterpackageDescription(*, fields="Built")

```

DESCRIPTIONVersionDESCRIPTION

```

package:utils2
packageVersion()if (packageVersion("MASS") < "7.3") { do.things }
packageDate()descpkglib.loc

```



```
DESCRIPTIONpackageDescription"packageDescription"drop = TRUE
DESCRIPTIONNAfieldsNA
packageVersion()"package_version"
packageDate()"Date"as.Date()NA
asDateBuilt(built)"Date"built
```

```
packageDate()date.fields
```

```
read.dcf
```

```
packageDescription("stats")
packageDescription("stats", fields = c("Package", "Version"))

packageDescription("stats", fields = "Version")
packageDescription("stats", fields = "Version", drop = FALSE)

if(requireNamespace("MASS") && packageVersion("MASS") < "7.3.29")
  message("you need to update 'MASS'")

pu <- packageDate("utils")
str(pu)
stopifnot(identical(pu, packageDate(desc = packageDescription("utils"))),
  identical(pu, packageDate("stats"))) # as "utils" and "stats" are
# both 'base R' and "Built" at same time
```

```
packageName
```

```
packageName(env = parent.frame())
```

```
env
```

```
envtopenv(env)envpackageName
packageNameNULL
```

```
NULL
```

```
getPackageName
```

```
packageName()
packageName(environment(mean))
```

packageStatus

```
packageStatus(lib.loc = NULL, repositories = NULL, method,
              type = getOption("pkgType"), ...)
```

```
## S3 method for class 'packageStatus'
summary(object, ...)
```

```
## S3 method for class 'packageStatus'
update(object, lib.loc = levels(object$inst$LibPath),
        repositories = levels(object$avail$Repository), ...)
```

```
## S3 method for class 'packageStatus'
upgrade(object, ask = TRUE, ...)
```

lib.loc	NULLNULL
repositories	contribNULL"repos"type
method	download.file
type	install.packages
object	"packageStatus"packageStatus
ask	TRUE
...	packageStatus available.packages installed.packages upgrade install.packages

```
printsummary"packageStatus"printsummary
update"packageStatus"upgradeupdate.packages
```

```
"packageStatus"
```

```
inst      installed.packages"Status"c("ok", "upgrade", "unavailable")
avail     available.packages"Status"c("installed", "not installed")
```

```
summary"summary.packageStatus"
```

```
Libs
Repos
```

[installed.packagesavailable.packages](#)

```
x <- packageStatus(repositories = contrib.url(findCRANmirror("web")))
print(x)
summary(x)

## Not run:
upgrade(x)
x <- update(x)
print(x)

## End(Not run)
```

page

[xfile.show](#)

```
page(x, method = c("dput", "print"), ...)
```

```
x
method          dputprint
...             dputprintfile.showtitle
```

```
xpage
titlefile.show...
```

[file.showeditfix](#)

[frame](#)

```
## Not run: ## four ways to look at the code of 'page'
page(page)          # as an object
page("page")       # a character string
v <- "page"; page(v) # a length-one character vector
page(utils::page)   # a call

## End(Not run)
```

person

```
person(given = NULL, family = NULL, middle = NULL,
       email = NULL, role = NULL, comment = NULL,
       first = NULL, last = NULL)

as.person(x)
## Default S3 method:
as.person(x)

## S3 method for class 'person'
format(x,
      include = c("given", "family", "email", "role", "comment"),
      braces = list(given = "", family = "", email = c("<", ">"),
                    role = c("[", "]"), comment = c("(", ")")),
      collapse = list(given = " ", family = " ", email = ", ",
                      role = ", ", comment = ", "),
      ...,
      style = c("text", "R", "md")
)

## S3 method for class 'person'
toBibtex(object, escape = FALSE, ...)

given
family
middle
email
role
comment
first
last
x          as.personas.person"person"
include
braces
collapse
...
style      "R" "md"
object     "person"
escape
```

```

"person"person()c()
format()includecollapsebracesbracescollapseFALSENAprint()format()
toBibtex()
$[
as.person()"and"
https://en.wikipedia.org/wiki/Personal\_name
"person"givenfamilyfirstmiddlelastgivenfamilyperson("R Core Team", role = "aut")
https://www.loc.gov/marc/relators/relaterm.html

"aut"
"com"
"cph"
"cre"
"ctb"
"ctr"
"dtc"
"fnd"
"rev"
"ths"
"trl"

"personList"personList()personList
comment"ORCID"https://orcid.org/print\(\)format\(\)"ROR"https://ror.org/
"cph"comment

```

```

person()as.person()"person"

```

citation

```

## Create a person object directly ...
p1 <- person("Karl", "Pearson", email = "pearson@stats.heaven")

## ... or convert a string.
p2 <- as.person("Ronald Aylmer Fisher")

## Combining and subsetting.
p <- c(p1, p2)
p[1]
p[-1]

## Extracting fields.
p$family
p$email

```

```

p[1]$email

## Specifying package authors, example from "boot":
## AC is the first author [aut] who wrote the S original.
## BR is the second author [aut], who translated the code to R [trl],
## and maintains the package [cre].
b <- c(person("Angelo", "Canty", role = "aut", comment =
  "S original, <http://statwww.epfl.ch/davison/BMA/library.html>"),
  person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
    comment = "R port", email = "ripley@stats.ox.ac.uk")
  )
b

## Formatting.
format(b)
format(b, include = c("family", "given", "role"),
  braces = list(family = c("", "", "role = c("(Role(s): ", ")")"))

## Conversion to BibTeX author field.
paste(format(b, include = c("given", "family")), collapse = " and ")
toBibtex(b)

## ORCID identifiers.
(p3 <- person("Achim", "Zeileis",
  comment = c(ORCID = "0000-0003-0918-3766")))

## ROR identifiers.
(p4 <- person("R Core Team",
  comment = c(ROR = "02zz1nj61")))

```

personList

[person](#)[personList](#)

personList(...)
as.personList(x)

...	"person"
x	as.person

["person"](#)

[person](#)

PkgUtils

R CMD check [options] pkgdirs
R CMD build [options] pkgdirs

pkgdirs checktar.tar.tar.gz.tgz.tar.bz2.tar.xz.tar.zstd
options

R CMD check
R CMD buildDESCRIPTION
R CMD --helpoptions
R CMD build_R_BUILD_RESAVE_DATA__R_BUILD_COMPACT_VIGNETTES_R CMD check
R CMD build"internal"tarR_BUILD_TAR
R CMD checkuntarR_INSTALL_TARR_INSTALL_TARtar.exeINSTALL

TMPDIRc:/TEMP

[RShowDoc](#)("R-exts")

process.events

process.events

process.events()

R_ProcessEvents

NULL

R_ProcessEvents

prompt

```
prompt(object, filename = NULL, name = NULL, ...)

## Default S3 method:
prompt(object, filename = NULL, name = NULL,
       force.function = FALSE, ...)

## S3 method for class 'data.frame'
prompt(object, filename = NULL, name = NULL, ...)

promptImport(object, filename = NULL, name = NULL,
             importedFrom = NULL, importPage = name, ...)

object          missingname
filename        name".Rd"NA
name
force.function  TRUEobject
...
importedFrom    objectobjectobject
importPage      object

filenameNAobjectfilenameman
filenameNACat(unlist(x), file = filename, sep = "\n")x
promptforname
importPagepromptImportapproxapproxfun.RdimportPage = "approxfun"

filenameNA

promptpromptData

prompt.data.frame

prompt.data.frame
```



```
promptDatahelpRShowDoc("R-exts")
```

```
package.skeleton
```

```
readline
```

```
require(graphics)
```

```
prompt(plot.default)
```

```
prompt(interactive, force.function = TRUE)
```

```
unlink("plot.default.Rd")
```

```
unlink("interactive.Rd")
```

```
prompt(women) # data.frame
```

```
unlink("women.Rd")
```

```
prompt(sunspots) # non-data.frame data
```

```
unlink("sunspots.Rd")
```

```
## Not run:
```

```
## Create a help file for each function in the .GlobalEnv:
```

```
for(f in ls()) if(is.function(get(f))) prompt(name = f)
```

```
## End(Not run)
```

promptData

```
promptData(object, filename = NULL, name = NULL)
```

```
object
```

```
filename      name".Rd"NA
```

```
name
```

```
filenameNAobjectfilename
```

```
filenameNACat(unlist(x), file = filename, sep = "\n")x
```

filenameNA

prompt

```
promptData(sunspots)
unlink("sunspots.Rd")
```

promptPackage

```
promptPackage(package, lib.loc = NULL, filename = NULL,
               name = NULL, final = FALSE)
```

```
package      character
lib.loc
filename      name".Rd"NA
name          "-package"\alias
final
```

```
filenameNApackagefilename
filenameNACat(unlist(x), file = filename, sep = "\n")x
finalTRUEfinal = TRUE
```

filenameNA

promptpackage.skeleton

```
filename <- tempfile()
promptPackage("utils", filename = filename)
file.show(filename)
unlink(filename)
```

Question

```
namehelp("name")?name
```

```
?topic
```

```
type?topic
```

```
topic
```

```
:::::topic
```

```
type package topic class topic type methods?method?
```

```
help
```

```
function if else for in repeat while break next reserved TRUE NA Inf
```

```
"?"
```

```
methods?~methods
```

```
topic type
```

```
"?" type method topic
```

```
selectMethod getMethod? package::generic()
```

```
help
```

```
??
```

```
?lapply
```

```
? "for" # but quotes/backticks are needed  
? ~+~
```

```
?women # information about data set "women"
```

```
package?parallel # overview help page of package 'parallel'
```

```
## Not run:  
require(methods)  
## define a S4 generic function and some methods
```

```

combo <- function(x, y) c(x, y)
setGeneric("combo")
setMethod("combo", c("numeric", "numeric"), function(x, y) x+y)

## assume we have written some documentation
## for combo, and its methods ....

?combo # produces the function documentation

methods?combo # looks for the overall methods documentation

method?combo("numeric", "numeric") # documentation for the method above

?combo(1:10, rnorm(10)) # ... the same method, selected according to
# the arguments (one integer, the other numeric)

?combo(1:10, letters) # documentation for the default method

## End(Not run)

```

rcompgen

```

rc.settings(ops, ns, args, dots, func, ipck, S3, data, help,
            argdb, fuzzy, quotes, files, backtick)

rc.status()
rc.getOption(name)
rc.options(...)

.DollarNames(x, pattern)
.AtNames(x, pattern)

## Default S3 method:
.DollarNames(x, pattern = "")
## S3 method for class 'list'
.DollarNames(x, pattern = "")
## S3 method for class 'environment'
.DollarNames(x, pattern = "")
## Default S3 method:
.AtNames(x, pattern = "")

findMatches(pattern, values, fuzzy, backtick)

```

```

ops          $@
ns

```

```

args
dots      ...args
func      "("
S3        args = TRUE
ipck      libraryrequire
data      data
help
argdb     args = TRUEargs...argdb = TRUE.addFunctionInfo
fuzzy     findMatches
backtick  findMatches
quotes    ?FALSEreadline
files     quotes
name...

        function.suffix "("
        funarg.suffix "="
        package.suffix ":@"
        options
x         "$"
pattern
values

rc.settingsrc.settingsipckfuncfuzzy
TRUE

$@ ops$@[$
    $@.DollarNames.AtNamesfindMatches
ns:::::foopkg:::foo"foopkg"
    MASMASS::MASS:::fr
help?classmethod
args+
    ...printplot
    S3
    seq([TAB]seq(from = 1, [TAB])seqseq(length[TAB]length.outlength(
    librarydataipcklibraryrequireinstalled.packagesinstalled.packages

findMatches.DollarNames.AtNamesfuzzy

```

```
rc.settingsNULL
rc.statusrc.status
```

```
comps          .completeToken
token          .assignToken.guessTokenFromLine
linebuffer     .assignLinebuffer
start          .assignStart
end            .assignEnd
fileName
fguess
isFirstArg
settingsoptions
rc.getOptionrc.optionsgetOptionoptions
findMatchesfuzzy
```

```
.assignToken(text)
.assignLinebuffer(line)
.assignStart(start)
.assignEnd(end)

.completeToken(custom = TRUE)
.retrieveCompletions()
.getFileComp()

.guessTokenFromLine()
.win32consoleCompletion(linebuffer, cursorPosition,
                        check.repeat = TRUE,
                        minlength = -1)

.addFunctionInfo(...)
```

```
textstartendline
.completeToken.retrieveCompletionsrc.options("custom.completer")NULLcustom    =
FALSE.completeToken
.getFileComp.completeTokenTRUE
.guessTokenFromLine
.win32consoleCompletionadditionpossiblecompsadditionpossiblecompspossiblewidth
comps
minlengthcheck.repeatTRUE

.addFunctionInfoargdbTRUE
```

```
ops = FALSEfoo@bafoofoo[i, 1:10]$bafoo[i, 1:10]
```

```
<deepayan.sarkar@r-project.org>
```

```
read.DIF
```

```
read.DIF(file, header = FALSE,  
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),  
  row.names, col.names, as.is = !stringsAsFactors,  
  na.strings = "NA", colClasses = NA, nrows = -1,  
  skip = 0, check.names = TRUE, blank.lines.skip = TRUE,  
  stringsAsFactors = FALSE,  
  transpose = FALSE, fileEncoding = "")
```

```
file  
      "clipboard"read.DIF("clipboard")  
header      headerTRUE  
dec  
numerals    type.convert  
row.names  
      row.names  
      row.names = NULL  
col.names   "V"  
as.is       colClasses  
            type.convertread.table  
            colClasses = "character"  
            as.is  
na.strings  NA  
colClasses  NA  
            NAtype.convert"NULL""factor""Date""POSIXct"as"character"  
            colClasses  
nrows  
skip  
check.names TRUEmake.names  
blank.lines.skip  
            TRUE  
stringsAsFactors  
  
transpose   transpose = TRUE  
fileEncoding file
```

```
data.framecol.namesheader = TRUE
```

```
as.iscolClasses  
colClasses
```

```
transpose
```

http://www.wotsit.org/https://en.wikipedia.org/wiki/Data_Interchange_Format

```
scantype.convertread.fwfreaddtabledata.frame
```

```
## read.DIF() may need transpose = TRUE for a file exported from Excel  
udir <- system.file("misc", package = "utils")  
dd <- read.DIF(file.path(udir, "exDIF.dif"), header = TRUE, transpose = TRUE)  
dc <- read.csv(file.path(udir, "exDIF.csv"), header = TRUE)  
stopifnot(identical(dd, dc), dim(dd) == c(4,2))
```

```
read.fortran
```

```
read.fortran(file, format, ..., as.is = TRUE, colClasses = NA)
```

```
file  
format  
...          read.fwf  
as.is  
colClasses   read.table
```

```
rFl.drDl.drXlrAlrIlldrFDAIXrdlXr  
format  
Xread.fwfcolClassescol.namesread.fwf
```



```
read.fortrand > 0FDd
```

[read.fwf](#)[read.table](#)[read.csv](#)

```
ff <- tempfile()
cat(file = ff, "123456", "987654", sep = "\n")
read.fortran(ff, c("F2.1", "F2.0", "I2"))
read.fortran(ff, c("2F1.0", "2X", "2A1"))
unlink(ff)
cat(file = ff, "123456AB", "987654CD", sep = "\n")
read.fortran(ff, list(c("2F3.1", "A2"), c("3I2", "2X")))
unlink(ff)
# Note that the first number is read differently than Fortran would
# read it:
cat(file = ff, "12.3456", "1234567", sep = "\n")
read.fortran(ff, "F7.4")
unlink(ff)
```

[read.fwf](#)

[data.frame](#)

```
read.fwf(file, widths, header = FALSE, sep = "\t",
         skip = 0, row.names, col.names, n = -1,
         buffersize = 2000, fileEncoding = "", ...)
```

file	file
widths	
header	sep
sep	
skip	read.table
row.names	read.table
col.names	read.table
n	
buffersize	
fileEncoding	file
...	read.table as.isna.stringscolClassesstrip.white

fileNA
-5read.tablecol.namescolClasses
buffersizebuffersize
read.fwfread.tableencoding

[data.frameread.table](#)

Perl

[scanread.table](#)

[read.fortran](#)

```
ff <- tempfile()
cat(file = ff, "123456", "987654", sep = "\n")
read.fwf(ff, widths = c(1,2,3))    #> 1 23 456 \\ 9 87 654
read.fwf(ff, widths = c(1,-2,3))    #> 1 456 \\ 9 654
unlink(ff)
cat(file = ff, "123", "987654", sep = "\n")
read.fwf(ff, widths = c(1,0, 2,3))  #> 1 NA 23 NA \\ 9 NA 87 654
unlink(ff)
cat(file = ff, "123456", "987654", sep = "\n")
read.fwf(ff, widths = list(c(1,0, 2,3), c(2,2,2))) #> 1 NA 23 456 98 76 54
unlink(ff)
```

read.socket

read.socketwrite.socket

read.socket(socket, maxlen = 256L, loop = FALSE)
write.socket(socket, string)

socket
maxlen
loop
string

read.socket
write.socket

`close.socket``make.socket`

```
finger <- function(user, host = "localhost", port = 79, print = TRUE)
{
  if (!is.character(user))
    stop("user name must be a string")
  user <- paste(user, "\r\n")
  socket <- make.socket(host, port)
  on.exit(close.socket(socket))
  write.socket(socket, user)
  output <- character(0)
  repeat{
    ss <- read.socket(socket)
    if (ss == "") break
    output <- paste(output, ss)
  }
  close.socket(socket)
  if (print) cat(output)
  invisible(output)
}
## Not run:
finger("root") ## only works if your site provides a finger daemon
## End(Not run)
```

`read.table`

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
  row.names, col.names, as.is = !stringsAsFactors, tryLogical = TRUE,
  na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = FALSE,
  fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
  dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
  dec = ",", fill = TRUE, comment.char = "", ...)
```

```
read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

file	<code>getwd()</code> <code>file</code> <code>file</code> <code>closest</code> <code>din()</code> <code>Ctrl-D</code> <code>Ctrl-Z</code> <code>stdin()</code> <code>file</code> <code>url</code>
header	<code>header</code> <code>TRUE</code>
sep	<code>sep = ""</code> <code>read.table</code>
quote	<code>quote = ""</code> <code>scan</code> <code>colClasses</code>
dec	
numerals	<code>type.convert</code>
row.names	<code>row.names</code> <code>row.names = NULL</code> <code>NULL</code> <code>row.names</code> <code>as.matrix</code>
col.names	<code>"v"</code>
as.is	<code>colClasses</code> <code>colClasses = "character"</code> <code>as.is</code>
tryLogical	<code>logical</code> <code>"F"</code> <code>"T"</code> <code>"FALSE"</code> <code>"TRUE"</code> <code>logical</code> <code>type.convert</code>
na.strings	<code>NA</code> <code>na.strings</code>
colClasses	<code>NA</code> <code>NA</code> <code>type.convert</code> <code>"NULL"</code> <code>"factor"</code> <code>"Date"</code> <code>"POSIXct"</code> <code>as</code> <code>"character"</code> <code>colClasses</code>
nrows	
skip	
check.names	<code>TRUE</code> <code>make.names</code>
fill	<code>TRUE</code>
strip.white	<code>sep</code> <code>character</code> <code>numeric</code> <code>scan</code>
blank.lines.skip	<code>TRUE</code>
comment.char	<code>""</code>
allowEscapes	<code>\</code> <code>scan</code>
flush	<code>TRUE</code> <code>scan</code>
stringsAsFactors	<code>as.is</code> <code>colClasses</code>
fileEncoding	<code>file</code>
encoding	<code>Encoding</code>
text	<code>file</code> <code>text</code>
skipNul	
...	<code>read.table</code>

```
colClassestype.convertas.is"42"
```

```
row.namesrow.names
```

```
col.namesfillblank.lines.skipcol.names
```

```
read.csvread.csv2read.table.csvread.csv2read.delimread.delim2header = TRUEfill =  
TRUE
```

```
blank.lines.skip = TRUE
```

```
skipNul = TRUE
```

```
data.frame
```

```
col.namesheader = TRUEcolClasses
```

```
encoding"latin1""UTF-8"
```

```
write.csv.csvread.csv(..., row.names = 1)
```

```
colClasses
```

```
nrows
```

```
comment.char = ""read.table
```

```
read.tablescan
```

```
as.iscolClasses
```

```
encodingfileEncoding
```

```
scantype.convertread.fwfwrite.tabledata.frame
```

```
count.fields
```

```
https://www.rfc-editor.org/rfc/rfc4180
```

```
## using count.fields to handle unknown maximum number of fields
## when fill = TRUE
test1 <- c(1:5, "6,7", "8,9,10")
tf <- tempfile()
writelines(test1, tf)

read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))
read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
unlink(tf)

## "Inline" data set, using text=
## Notice that leading and trailing empty lines are auto-trimmed

read.table(header = TRUE, text = "
a b
1 2
3 4
")
```

readRegistry

```
readRegistry(key, hive = c("HLM", "HCR", "HCU", "HU", "HCC", "HPD"),
            maxdepth = 1, view = c("default", "32-bit", "64-bit"))
```

key

hive HKEY_LOCAL_MACHINE\HKEY_CLASSES_ROOT\HKEY_CURRENT_USER\HKEY_USERS
 HKEY_CURRENT_CONFIG\HKEY_PERFORMANCE_DATA

maxdepth

view

NULL

<https://learn.microsoft.com/en-us/windows/win32/winprog64/registry-redirector>

```

if(.Platform$OS.type == "windows") withAutoprint({
  ## only in HLM if set in an admin-mode install.
  try(readRegistry("SOFTWARE\\R-core", maxdepth = 3))

  gmt <- file.path("SOFTWARE", "Microsoft", "Windows NT",
                  "CurrentVersion", "Time Zones",
                  "GMT Standard Time", fsep = "\\")
  readRegistry(gmt, "HLM")
})
## Not run: ## on a 64-bit R need this to find 32-bit JAGS
readRegistry("SOFTWARE\\JAGS", maxdepth = 3, view = "32")

## See if there is a 64-bit user install
readRegistry("SOFTWARE\\R-core\\R64", "HCU", maxdepth = 2)

## End(Not run)

```

recover

```
options(error = recover)
```

```
recover()
```

```
recoverbrowser
```

```
crecover0recover
```

```
recoverdump.framesrecoverdump.framesrecoverdump.frames
```

```
recover0recover
```

```
recoverupdown
```

```
browseroptionsdump.frames
```

```

## Not run:

options(error = recover) # setting the error option

### Example of interaction

> myFit <- lm(y ~ x, data = xy, weights = w)
Error in lm.wfit(x, y, w, offset = offset, ...) :
  missing or negative weights not allowed

Enter a frame number, or 0 to exit
1:lm(y ~ x, data = xy, weights = w)
2:lm.wfit(x, y, w, offset = offset, ...)
Selection: 2
Called from: eval(expr, envir, enclos)
Browse[1]> objects() # all the objects in this frame
[1] "method" "n"      "ny"      "offset" "tol"    "w"
[7] "x"      "y"
Browse[1]> w
[1] -0.5013844  1.3112515  0.2939348 -0.8983705 -0.1538642
[6] -0.9772989  0.7888790 -0.1919154 -0.3026882
Browse[1]> dump.frames() # save for offline debugging
Browse[1]> c # exit the browser

Enter a frame number, or 0 to exit
1:lm(y ~ x, data = xy, weights = w)
2:lm.wfit(x, y, w, offset = offset, ...)
Selection: 0 # exit recover
>

## End(Not run)

```

relist

```
relist()unlist(obj)obj"relistable"
```

```

relist(flesh, skeleton)
## Default S3 method:
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'factor'
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'list'
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'matrix'
relist(flesh, skeleton = attr(flesh, "skeleton"))

as.relistable(x)
is.relistable(x)

```



```
## S3 method for class 'relistable'
unlist(x, recursive = TRUE, use.names = TRUE)
```

```
flesh
skeleton
x
recursive      x
use.names
```

```
optimnmlmunlist()relist()optim()
```

```
list(mean = c(0, 1), vcov = cbind(c(1, 1), c(1, 0))).
```

```
optimmvdnorm(x, mean, vcov, log = FALSE)
```

```
ipar <- list(mean = c(0, 1), vcov = c bind(c(1, 1), c(1, 0)))
initial.param <- as.relistable(ipar)
```

```
ll <- function(param.vector)
{
  param <- relist(param.vector, skeleton = ipar)
  -sum(mvdnorm(x, mean = param$mean, vcov = param$vcov,
              log = TRUE))
}
```

```
optim(unlist(initial.param), ll)
```

```
relistshapeflesh
```

```
relist(flesh, skeleton)
```

```
unlist(as.relistable(obj))optimllpar
```

```
skeletonunlist
```

```
relist(unlist(x), x) == x
unlist(relist(y, skeleton)) == y
```

```
x <- as.relistable(x)
relist(unlist(x)) == x
```

```
NULL
```

```
"relistable""list"
```

[unlist](#)

```
ipar <- list(mean = c(0, 1), vcov = cbind(c(1, 1), c(1, 0)))
initial.param <- as.relistable(ipar)
ul <- unlist(initial.param)
relist(ul)
stopifnot(identical(relist(ul), initial.param))
```

REMOVE

R CMD REMOVE [options] [-l lib] pkgs

pkgs
lib --library=lib
options

R CMD REMOVE pkgslib
R CMD REMOVE -l lib
R CMD REMOVE --help

REMOVED-devel

[INSTALLremove.packages](#)

remove.packages

remove.packages(pkgs, lib)

pkgs
lib [.libPaths\(\)](#)

[REMOVE](#)

[install.packages](#)

removeSource

```
options("keep.source")TRUEparse()removeSource()
```

```
removeSource(fn)
```

```
fn          functionis.language
```

```
"srcref"body(fn)
```

```
fn
```

```
is.language
```

```
srcrefdeparse
```

```
## to make this act independently of the global 'options()' setting:
op <- options(keep.source = TRUE)
fn <- function(x) {
  x + 1 # A comment, kept as part of the source
}
fn
names(attributes(fn))      # "srcref" (only)
names(attributes(body(fn))) # "srcref" "srcfile" "wholeSrcref"
f2 <- removeSource(fn)
f2
stopifnot(length(attributes(fn)) > 0,
           is.null(attributes(f2)),
           is.null(attributes(body(f2))))

## Source attribute of parse()d expressions,
## have {"srcref", "srcfile", "wholeSrcref"} :
E <- parse(text ="a <- x^y # power") ; names(attributes(E ))
E. <- removeSource(E)                ; names(attributes(E.))
stopifnot(length(attributes(E )) > 0,
           is.null(attributes(E.)))
options(op) # reset to previous state
```

RHOME

R RHOME

roman

```
as.roman(x)
.romans
```

```
r1 + r2
r1 <= r2
max(r1)
sum(r2)
```

```
x
r1r2          class"roman"
```

```
as.roman"roman"methods(class = "roman")
"Arith""Compare""Logic""Ops"
as.roman(NA)
.romanscharacter
```

https://en.wikipedia.org/w/index.php?title=Roman_numerals&oldid=1188781837

```
## First five roman 'numbers'.
(y <- as.roman(1 : 5))
## Middle one.
y[3]
## Current year as a roman number.
(y <- as.roman(format(Sys.Date(), "%Y")))
## Today, and 10, 20, 30, and 100 years ago ...
y - 10*c(0:3,10)

## mixture of arabic and roman numbers :
as.roman(c(NA, 1:3, "", strrep("I", 1:7))) # + NA with a warning for the last.
cc <- c(NA, 1:3, strrep("I", 0:6)) # "IIIIII" is historical (Wikipedia)
(rc <- as.roman(cc)) # two NAs: 0 is not "roman"
(ic <- as.integer(rc)) # works automatically [without an explicit method]
rNA <- as.roman(NA)
## simple consistency checks -- arithmetic when result is in {1,2,..,4999} :
stopifnot(identical(rc, as.roman(rc)), # as.roman(.) is "idempotent"
           identical(rc + rc + (3*rc), rc*5),
           identical(ic, c(NA, 1:3, NA, 1:6)),
           identical(as.integer(5*rc), 5L*ic),
           identical(as.numeric(rc), as.numeric(ic)),
```

```

        identical(rc[1], rNA),
        identical(as.roman(0), rNA),
        identical(as.roman(NA_character_), rNA),
        identical(as.list(rc), as.list(ic)))
## Non-Arithmetic 'Ops' :
stopifnot(exprs = {
  # Comparisons :
  identical(ic <= (ii <- c(0:4,1:6)), rc <= ii)
  identical(ic == ii, rc == as.roman(ii))
  # Logic [integers |>- logical] :
  identical(rc & TRUE , ic & TRUE)
  identical(rc & FALSE, ic & FALSE)
  identical(rc | FALSE, ic | FALSE)
  identical(rc | NA   , ic | NA)
})
## 'Summary' group functions (and comparison):
(rc. <- rc[!is.na(rc)])
stopifnot(exprs = {
  identical(min(rc), as.roman(NA))
  identical(min(rc, na.rm=TRUE),
    as.roman(min(ic, na.rm=TRUE)))
  identical(range(rc.),
    as.roman(range(as.integer(rc.))))
  identical(sum (rc, na.rm=TRUE), as.roman("XXVII"))
  identical(format(prod(rc, na.rm=TRUE)), "MMMCCCXX")
    format(prod(rc.)) == "MMMCCCXX"
  identical(format(prod(rc[-11], na.rm=TRUE)), "DCCXX")
})

```

Rprof

```

Rprof(filename = "Rprof.out", append = FALSE, interval = 0.02,
  memory.profiling = FALSE, gc.profiling = FALSE,
  line.profiling = FALSE, filter.callframes = FALSE,
  numfiles = 100L, bufsize = 10000L,
  event = c("default", "cpu", "elapsed"))

```

```

filename      NULL""
append
interval
memory.profiling

```

```

gc.profiling
line.profiling
filter.callframes

```

```

numfilesbufsize

```

```

event          "elapsed""cpu""default"

```

```
intervalsummaryRprofR CMD RprofR CMD Rprof --help
```

```
"elapsed"readline()
"cpu"system"elapsed"
"cpu"
"default""elapsed""cpu"
"elapsed"/etc/security/limits.conf
sys.calls"special"
line.profilingTRUEparsesummaryRprof
```

```
try(EXPR)EXPR
```

```
1. +-base::try(EXPR)
2. | \-base::tryCatch(...)
3. |   \-base::tryCatchList(expr, classes, parentenv, handlers)
4. |     \-base::tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. |       \-base::doTryCatch(return(expr), name, parentenv, handler)
6. \-EXPR
```

```
EXPRfilter.callframesTRUE
```

```
eval()eval()
```

```
calling <- function() evaluator(quote(called()), environment())
evaluator <- function(expr, env) eval(expr, env)
called <- function() EXPR()
```

```
calling()called()eval()evaluator()called()
```

```
1. calling()
2. \-evaluator(quote(called()), environment())
3.   \-base::eval(expr, env)
4.     \-base::eval(expr, env)
5.       \-called()
6.         \-EXPR()
```

```
called()
```

```
1. calling()
5. \-called()
6.   \-EXPR()
```

```
eval()
```

```
calling <- function() evaluator(quote(called()), new.env())
```

```
5. called()
6. \-EXPR()
```

```
--disable-R-profiling
    Rprof(event = "cpu")-p-pg
filename
numfilesbufsizeRprof(NULL)
```

```
RShowDoc("R-exts")
summaryRprof
tracememRprofmem
```

```
## Not run: Rprof()
## some code to be profiled
Rprof(NULL)
## some code NOT to be profiled
Rprof(append = TRUE)
## some code to be profiled
Rprof(NULL)
## ...
## Now post-process the output as described in Details

## End(Not run)
```

Rprofmem

```
Rprofmem(filename = "Rprofmem.out", append = FALSE, threshold = 0)
```

```
filename      NULL""
append
threshold
```

```
mallocmalloc
```

Rprof
tracemem

```
## Not run:
## not supported unless R is compiled to support it.
Rprofmem("Rprofmem.out", threshold = 1000)
example(glm)
Rprofmem(NULL)
noquote(readLines("Rprofmem.out", n = 5))

## End(Not run)
```

Rscript

#!

Rscript [options] file [args]
Rscript [options] -e expr [-e expr2 ...] [args]

options	--
expr	expr2
file	-stdin
args	file-e

Rscript --helpRscript --versionRscript
#!file--no-echo --no-restore--no-save####
-efile-e
--verbose-eargs-efile
optionsfile-e
-verbose Rscript
-default-packages=list listNULLR_DEFAULT_PACKAGES
exprfile
--default-packagesRscriptR_SCRIPT_DEFAULT_PACKAGESR_DEFAULT_PACKAGES
RHOME
stdin()file("stdin")stdin

Rscriptexecv

```
## Not run:
Rscript -e 'date()' -e 'format(Sys.time(), "%a %b %d %X %Y")'

# Get the same initial packages in the same order as default R:
Rscript --default-packages=methods,datasets,utils,grDevices,graphics,stats -e 'sessionInfo()'

## example #! script for a Unix-alike
## (arguments given on the #! line end up as [options] to Rscript, while
## arguments passed to the #! script end up as [args], so available to
## commandArgs())
#! /path/to/Rscript --vanilla --default-packages=utils
args <- commandArgs(TRUE)
res <- try(install.packages(args))
if(inherits(res, "try-error")) q(status=1) else q()

## End(Not run)
```

RShowDoc

RShowDoc(what, type = c("pdf", "html", "txt"), package)

what
type
package

whatR-adminR-dataR-extsR-introR-intsR-langNEWSCOPYINGshare/licensesFAQR-FAQ/doc
rw-FAQ
packagedoc
what
type"pdf"

[help?help.start](#)

type = "txt"[file.showvignette](#)RShowDoc(*, package= .)

```
RShowDoc("R-lang")
RShowDoc("FAQ", type = "html")
RShowDoc("frame", package = "grid")
RShowDoc("changes.txt", package = "grid")
RShowDoc("NEWS", package = "MASS")
```

RSiteSearch

<https://search.r-project.org>

```
RSiteSearch(string,
  restrict = c("functions", "descriptions", "news", "Rfunctions",
    "Rmanuals", "READMEs", "views", "vignettes"),
  format,
  sortby = c("score", "date:late", "date:early", "subject",
    "subject:descending", "size", "size:descending"),
  matchesPerPage = 20,
  words = c("all", "any"))
```

string	
restrict	functionsdescriptionsnewsRfunctionsRmanualsREADMEsREADMEviews vignettes
format	
sortby	scoredate:latedate:earlysubjectsubject:descendingsize size:descending
matchesPerPage	
words	allany

<https://search.r-project.org>

[help.searchhelp.start](#)
[browseURL](#)

```
# need Internet connection
## for phrase searching you may use (escaped) double quotes or brackets
RSiteSearch("{logistic regression} \"glm object\")
RSiteSearch('logistic regression')

## Search in vignettes and help files of R base packages
## store the query string:
fullquery <- RSiteSearch("lattice", restrict = c("vignettes","Rfunctions"))
fullquery # a string of 112 characters
```

rtags

rtags

```
rtags(path = ".", pattern = "\\.[RrSs]$",
      recursive = FALSE,
      src = list.files(path = path, pattern = pattern,
                      full.names = TRUE,
                      recursive = recursive),
      keep.re = NULL,
      ofile = "", append = FALSE,
      verbose = getOption("verbose"),
      type = c("etags", "ctags"))
```

pathpatternrecursive

`list.files.R.r.S.ssrc`

src

keep.re srckkeep.re = "/R/[^/]*\\.R\$".RR

ofile `catfile`"TAGS""tags"

append

verbose TRUE

type "etags""ctags"

ctagsetags

rtags

R CMD rtags

<https://en.wikipedia.org/wiki/Ctags>https://www.gnu.org/software/emacs/manual/html_node/emacs/Tags-Tables.html

[list.filescat](#)

```
## Not run:
rtags("/path/to/src/repository",
      pattern = "[.]*\\. [RrSs]$",
      keep.re = "/R/",
      verbose = TRUE,
      ofile = "TAGS",
      append = FALSE,
      recursive = TRUE)

## End(Not run)
```

Rtangle

Stangle

[Stangle](#)[RtangleSetup\(\)](#)[Stangle\(\)](#)

```
Rtangle()
RtangleSetup(file, syntax, output = NULL, annotate = TRUE,
             split = FALSE, quiet = FALSE, drop.evalFALSE = FALSE, ...)
```

file	Sweave
syntax	SweaveSyntax
output	split = TRUE
annotate	function FALSEannotate
split	
quiet	
drop.evalFALSE	eval = FALSEdrop.evalFALSE = TRUE
...	
split = TRUEbasename(file)RnwStexR"stdout""stderr"	
.R	

annotate

```
annotate = TRUE

#####
### code chunk number 3: viewport
#####

#####
### code chunk number 18: grid.Rnw:647-648
#####

#####
### code chunk number 19: trellisdata (eval = FALSE)
#####
```

```
annotate(options, chunk, output)Rtangle()$runcode
```

Rtangle

```
"R"engine"R""S"
TRUEkeep.source == TRUE
TRUEFALSE
split = TRUEprefix.string
split = TRUEprefix = TRUE
FALSE
```

[SweaveRweaveLatex](#)

```
nmRnw <- "example-1.Rnw"
exfile <- system.file("Sweave", nmRnw, package = "utils")
## Create R source file
Stangle(exfile)
nmR <- sub("Rnw$", "R", nmRnw) # the (default) R output file name
if(interactive()) file.show("example-1.R")

## Smaller R source file with custom annotation:
my.Ann <- function(options, chunk, output) {
  cat("### chunk #", options$chunknr, ": ",
      if(!is.null(ol <- options$label)) ol else .RtangleCodeLabel(chunk),
      if(!options$eval) " (eval = FALSE)", "\n",
      file = output, sep = "")
}
```

```

Stangle(exfile, annotate = my.Ann)
if(interactive()) file.show("example-1.R")

Stangle(exfile, annotate = my.Ann, drop.evalFALSE=TRUE)
if(interactive()) file.show("example-1.R")

```

RweaveLatex

[Sweaveparse\(\)](#)[eval\(\)](#)

RweaveLatex()

```

RweaveLatexSetup(file, syntax, output = NULL, quiet = FALSE,
                  debug = FALSE, stylepath, ...)

```

file	Sweave
syntax	SweaveSyntax
output	.nw.Rnw.Snw.texfile
quiet	TRUE
debug	TRUE
stylepath	
...	

```

\usepackage{Sweave}RweaveLatex\begin{document}stylepath = TRUESweave.stySweave
Sweave.sty
stylepathSWEAVE_STYLEPATH_DEFAULTFALSETRUEFALSEFALSE
Sweave.sty/share/texmf
Sweave.stygraphicx
\setkeys{Gin}{width=0.8\textwidth}
graphicx\includegraphics{}
width=30.8\textwidth
\setkeys{Gin}{width=...}.Rnw\begin{document}width
\usepackage[nogin]{Sweave}\includegraphics{}heightwidth
Sweave.sty[nofontenc]\usepackage[T1]{fontenc}
[inconsolata]inconsolata
sQuoteoptions(useFancyQuotes = FALSE)
.png.jpg\DeclareGraphicsExtensions{.png,.pdf,.jpg}pdfpngTRUESweave
DeclareGraphicsExtensionspdflatex

```

RweaveLatexSweave...

"R"engine"R""S"

TRUE

TRUETRUE

TRUEFALSE

"verbatim""verbatim"Soutput"tex""hide"

FALSETRUE

TRUETRUEFALSEprint

FALSETRUE

"true""true""all""false"

TRUETRUEprefix.string

TRUEsplit = TRUE\includegraphicsinclude = FALSE

FALSE

FALSEfig = FALSE

TRUEfig = FALSE

pdfpdf.options()

FALSEfig = FALSE \geq 2.13.0

FALSEfig = FALSE \geq 2.13.0

NULLfig = FALSE \geq 2.13.0

100

FALSE

FALSE

\SweaveOpts{}

grdevicefigeval

get(options\$grdevice, envir = .GlobalEnv)(name=, width=,
height=, options)

dev.off

<<results=hide>>=

my.Swd <- function(name, width, height, ...)
 grDevices::png(filename = paste(name, "png", sep = "."),
 width = width, height = height, res = 100,
 units = "in", type = "quartz", bg = "transparent")

@

.GlobalEnvgrdevice = "pkg::my.Swd":::::

dev.off.offmy.Swd.off()pkg::my.Swd.off()

```
getOption("SweaveHooks")echoprintTRUEgetOption("SweaveHooks")"SweaveHooks"  
list(fig = foo)foo  
cleanclean = TRUE
```

SweaveRtangle

Rwin configuration

```
RconsoleRgui loadRconsole(*)  
Rdevgawindowswin.graphwin.metafilewin.printbmpjpegpnggtifftype = "windows"  
windows
```

```
loadRconsole(file)
```

```
file Rconsole
```

```
\etcR_USER
```

```
R_USERHOME{HOMEDRIVE}{HOMEPATH}HOMEDRIVEHOMEDRIVERw-FAQ
```

```
RdevgaRconsoleoptions("width")
```

```
Rconsole
```

```
Rconsole
```

```
loadRconsole
```

```
MS MinchoArial
```

```
GUI preferencesEdit
```


windows

```
if(.Platform$OS.type == "windows") withAutoprint({
  ruser <- Sys.getenv("R_USER")
  cat("\n\nLocation for personal configuration files is\n  R_USER = ",
      ruser, "\n\n", sep = "")
  ## see if there are personal configuration files
  file.exists(file.path(ruser, c("Rconsole", "Rdevga"))))

## show the configuration files used
showConfig <- function(file)
{
  ruser <- Sys.getenv("R_USER")
  path <- file.path(ruser, file)
  if(!file.exists(path)) path <- file.path(R.home(), "etc", file)
  file.show(path, header = path)
}
showConfig("Rconsole")
})
```

savehistory

```
loadhistory(file = ".Rhistory")
savehistory(file = ".Rhistory")

history(max.show = 25, reverse = FALSE, pattern, ...)

timestamp(stamp = date(),
           prefix = "##----- ", suffix = " -----##",
           quiet = FALSE)
```

```
file
max.show      Inf
reverse
pattern
...           grep
stamp
prefix
suffix
quiet        TRUE
```

```
RguiRtermRterm
readlinereadline
  R.app.Rapp.history

readlineR_HISTSIZER_HISTFILE.RhistoryloadhistorysavehistoryR_HISTSIZ
Sys.setenv
readline0600
timestamp
```

```
savehistory().Last
```

```
## Not run:
## Save the history in the home directory: note that it is not
## (by default) read from there but from the current directory
.Last <- function()
  if(interactive()) try(savehistory("~/Rhistory"))

## End(Not run)
```

```
select.list
```

```
select.list(choices, preselect = NULL, multiple = FALSE,
            title = NULL, graphics = getOption("menu.graphics"))
```

```
choices
preselect      NULL
multiple
title          NULL
graphics
```

```
graphics = TRUE
multiple
```

```
graphicsmultiple = FALSEmenumultiple = TRUE"+"
select.list
```

```
multipleCancel""multipleCancel
```

```
menutk_select.list
```

```
## Not run:  
select.list(sort(.packages(all.available = TRUE)))  
  
## End(Not run)
```

```
sessionInfo
```

```
print()toLatex()"sessionInfo"localetzonesystem.codepagecode.page110n_info()
```

```
sessionInfo(package = NULL)  
## S3 method for class 'sessionInfo'  
print(x, locale = TRUE, tzone = locale,  
      RNG = !identical(x$RNGkind, .RNGdefaults), ...)  
## S3 method for class 'sessionInfo'  
toLatex(object, locale = TRUE, tzone = locale,  
        RNG = !identical(object$RNGkind, .RNGdefaults), ...)  
osVersion
```

```
package      NULL  
x            "sessionInfo"  
object       "sessionInfo"  
locale       tzone  
tzone  
RNG          RNGkind()RNGversion(*)  
...
```

```
sessionInfo()"sessionInfo"printtoLatex
```

```
R.version    R.Version()  
platform     platform/sub-arch(32-bit)  
running      NULLosVersion  
RNGkind      RNGkind()  
matprod      getOption("matprod")  
BLAS         extSoftVersion()["BLAS"]
```

LAPACK	La_library()
LA_version	La_version()
locale	Sys.getlocale()
tzone	Sys.timezone()
tzcode_type	
basePkgs	
otherPkgs	packageDescription
loadedOnly	packageDescription

osVersion

```
osVersionNULL
sessionInfo()$running
win.version
osVersionsessionInfo()$running
10.161112
```

[R.versionR_compiled_by](#)

```
sI <- sessionInfo()
sI
# The same, showing the RNGkind, but not the locale :
print(sI, RNG = TRUE, locale = FALSE)
toLatex(sI, locale = FALSE) # shortest; possibly desirable at end of report
```

setRepositories

```
setRepositories(graphics = getOption("menu.graphics"),
               ind = NULL, addURLs = character(), name = NULL)
```

graphics	menu
ind	NULLgraphics = FALSE
name	NULLind
addURLs	

```
/etc/repositoriesR_REPOSITORIESNULL/.R/repositories
options("BioC_mirror")chooseBioCmirror"https://bioconductor.org"R_BIOC_VERSION
options("repos")
"pkgType""both""source"
CRANgetOption("repos")"CRAN"
indname
```

```
options("repos")reposlistreposNULL
```

```
options(repos =)repositories
```

```
chooseCRANmirrorchooseBioCmirrorinstall.packages
```

```
## Not run:
setRepositories(addURLs =
  c(CRANxtras = "https://www.stats.ox.ac.uk/pub/RWin"))

## End(Not run)
oldrepos <- setRepositories(name = c("CRAN", "R-Forge"))
getOption("repos")
options(oldrepos) # restore
```

```
setWindowTitle
```

```
RGui
```

```
setWindowTitle(suffix, title = paste(getIdentification(), suffix))
```

```
getWindowTitle()
```

```
getIdentification()
```

```
setStatusbar(text)
```

```
suffix
```

```
title
```

```
text
```

```
setWindowTitlesuffixRGuiR ConsoleRtermsuffix = ""
getWindowTitle
RGui --sdiRterm
getIdentification
setStatusBar
```

```
setWindowTitle
getWindowTitlegetIdentification
```

```
RguiRtermESSRStudio""
```

```
if(.Platform$OS.type == "windows") withAutoprint({
## show the current working directory in the title, saving the old one
oldtitle <- setWindowTitle(getwd())
Sys.sleep(0.5)
## reset the title
setWindowTitle("")
Sys.sleep(0.5)
## restore the original title
setWindowTitle(title = oldtitle)
})
```

SHLIB

```
dyn.loadlibrary.dynam
```

```
R CMD SHLIB [options] [-o dllname] files
```

```
files
dllname      .so.dll
options      R CMD SHLIB --help
```

```
R CMD SHLIBINSTALL
```

```
cpp
files.c.cpp.cc.C.f.f90.f95.m.M.mm.o
```

```
-n--dry-run
```

SHLIBR-devel

COMPILEdyn.loadlibrary.dynam

```
## Not run:  
# To link against a library not on the system library paths:  
R CMD SHLIB -o mylib.so a.f b.f -L/opt/acml3.5.0/gnu64/lib -lacml  
  
## End(Not run)
```

shortPathName

GetShortPathNameW

shortPathName(path)

path

\

normalizePath

```
if(.Platform$OS.type == "windows") withAutoprint({  
  cat(shortPathName(c(R.home(), tempdir()))), sep = "\n")  
})
```

sourceutils

```
getSrcFilename(x, full.names = FALSE, unique = TRUE)
getSrcDirectory(x, unique = TRUE)
getSrcCref(x)
getSrcLocation(x, which = c("line", "column", "byte", "parse"),
               first = TRUE)
```

```
x
full.names
unique
which
first
```

```
"keep.source"TRUE
```

```
"column"
"line"#line
```

```
getSrcFilenamegetSrcDirectory
getSrcCref"srcCref"NULL
getSrcLocation
```

```
srcCrefgetParseData
```

```
fn <- function(x) {
  x + 1 # A comment, kept as part of the source
}
```

```
# Show the temporary file directory
# where the example was saved
```

```
getSrcDirectory(fn)
getSrcLocation(fn, "line")
```

stack

```
stack(x, ...)
## Default S3 method:
stack(x, drop=FALSE, ...)
## S3 method for class 'data.frame'
stack(x, select, drop=FALSE, ...)
```

```
unstack(x, ...)
## Default S3 method:
unstack(x, form, ...)
## S3 method for class 'data.frame'
unstack(x, form, ...)
```

```
x
select
form          formula(x)unstack
drop
...
```

```
stackunstack
stackis.vectorunlist
as.list
```

```
unstackform
stack
values      x
ind         x
```

[lmreshape](#)

```
require(stats)
formula(PlantGrowth) # check the default formula
pg <- unstack(PlantGrowth) # unstack according to this formula
pg
stack(pg) # now put it back together
stack(pg, select = -ctrl) # omitting one vector
```

str

[summarydputargs](#)

strOptions()[options](#)(str = .)

str(object, ...)

S3 method for class 'data.frame'

str(object, ...)

Default S3 method:

```
str(object, max.level = NA,
     vec.len = str0$vec.len, digits.d = str0$digits.d,
     nchar.max = 128, give.attr = TRUE,
     drop.deparse.attr = str0$drop.deparse.attr,
     give.head = TRUE, give.length = give.head,
     width = getOption("width"), nest.lev = 0,
     indent.str = paste(rep.int(" ", max(0, nest.lev + 1)),
                        collapse = ".."),
     comp.str = "$ ", no.list = FALSE, envir = baseenv(),
     strict.width = str0$strict.width,
     formatNum = str0$formatNum, list.len = str0$list.len,
     deparse.lines = str0$deparse.lines, ...)
```

```
strOptions(strict.width = "no", digits.d = 3, vec.len = 4,
           list.len = 99, deparse.lines = NULL,
           drop.deparse.attr = TRUE,
           formatNum = function(x, ...)
             format(x, trim = TRUE, drop0trailing = TRUE, ...))
```

object

max.level

vec.len vec.len"str"[options](#)

digits.d [print](#)digits.d"str"

nchar.max [character](#)longch

give.attr TRUE

drop.deparse.attr

TRUE[deparse](#)(control =)"showAttributes"FALSEstrOptions()

give.length TRUE[1:...]

give.head TRUE[1:...]

width [options](#)("width")strict.width"no"

nest.lev str

indent.str

```

comp.str
no.list
envir          delayedAssign
strict.width   widthc("no", "cut", "wrap")strict.width"str"options"no""wrap"
               strwrap(*, width = width)"cut"widthvec.lengthstrict.width =
               "wrap"

formatNum      formatformatNum"str"strOptions()formatC
list.len
deparse.lines  NULLnlinesdeparse()objectcallNULLnchar.maxwidth
...

```

```
str
```

```

objectclassstr()""["length()"is.list(object)TRUElength(object)
length(unclass(object))unclass(object)

```

```
<maechler@stat.math.ethz.ch>
```

```
ls.strsummaryargs
```

```

require(stats); require(grDevices); require(graphics)
## The following examples show some of 'str' capabilities
str(1:12)
str(ls)
str(args) #- more useful than  args(args) !
str(freeny)
str(str)
str(.Machine, digits.d = 20) # extra digits for identification of binary numbers
str( lsfit(1:9, 1:9))
str( lsfit(1:9, 1:9), max.level = 1)
str( lsfit(1:9, 1:9), width = 60, strict.width = "cut")
str( lsfit(1:9, 1:9), width = 60, strict.width = "wrap")
op <- options(); str(op)   # save first;
                           # otherwise internal options() is used.

need.dev <-
  !exists(".Device") || is.null(.Device) || .Device == "null device"
{ if(need.dev) pdf()
  str(par())
  if(need.dev) graphics.off()
}
ch <- letters[1:12]; is.na(ch) <- 3:5
str(ch) # character NA's

str(list(a = "A", L = as.list(1:100)), list.len = 9)
##          -----

```

```
## " .. [list output truncated] "

## Long strings, 'nchar.max'; 'strict.width' :
nchar(longch <- paste(rep(letters,100), collapse = ""))
str(longch)
str(longch, nchar.max = 52)
str(longch, strict.width = "wrap")

## Multibyte characters in strings:
## Truncation behavior (<-> correct width measurement) for "long" non-ASCII:
idx <- c(65313:65338, 65345:65350)
fwch <- intToUtf8(idx) # full width character string: each has width 2
ch <- strtrim(paste(LETTERS, collapse="._"), 64)
(ncc <- c(c.ch = nchar(ch), w.ch = nchar(ch, "w"),
          c.fw = nchar(fwch), w.fw = nchar(fwch, "w")))
stopifnot(unname(ncc) == c(64,64, 32, 64))
## nchar.max: 1st line needs an increase of 2 in order to see 1 (in UTF-8!):
invisible(lapply(60:66, function(N) str(fwch, nchar.max = N)))
invisible(lapply(60:66, function(N) str( ch , nchar.max = N))) # "1 is 1" here

## Settings for narrow transcript :
op <- options(width = 60,
              str = strOptions(strict.width = "wrap"))
str(lsf1(1:9,1:9))
str(options())
## reset to previous:
options(op)

str(quote( { A+B; list(C, D) } ))

## S4 classes :
require(stats4)
x <- 0:10; y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)
ll <- function(ymax = 15, xh = 6)
  -sum(dpois(y, lambda=ymax/(1+x/xh), log=TRUE))
fit <- mle(ll)
str(fit)
```

```
strcapture
```

```
strcapture
```

```
strcapture(pattern, x, proto, perl = FALSE, useBytes = FALSE)
```

```
pattern
```

```
x
proto          data.frame
perluseBytes   regexec
```

```
protodata.frame
```

```
protodata.frameprotoxpathpatternNA
```

[regexec](#)[regmatches](#)

```
x <- "chr1:1-1000"
pattern <- "(.*?):([[:digit:]]+)-([[:digit:]]+)"
proto <- data.frame(chr=character(), start=integer(), end=integer())
strcapture(pattern, x, proto)
```

```
summaryRprof
```

[Rprof](#)

```
summaryRprof(filename = "Rprof.out", chunksize = 5000,
              memory = c("none", "both", "tseries", "stats"),
              lines = c("hide", "show", "both"),
              index = 2, diff = TRUE, exclude = NULL,
              basenames = 1)
```

```
filename      Rprof()
chunksize
memory
lines
index
diff          TRUE
exclude
basenames
```

[Rprof](#)[R CMD Rprof](#)
chunksizechunksize

```
memory = "none"lines = "hide"
```

```
by.self
```

```
by.total
```

```
sample.interval
```

```
sampling.time
```

```
self.timeself.pcttotal.timetotal.pct
```

```
lines = "show"
```

```
by.line
```

```
memory = "both"
```

```
memory = "tseries"
```

```
memory = "stats"by
```

```
memory = "none"Rprof(memory.profiling = TRUE)
```

```
memory.profiling = TRUEallocduplicateduplicate
```

```
memory = "both"
```

```
memory = "tseries"memory = "stats"indexmemory = "tseries"memory = "stats"memory  
= "tseries"memory = "stats"diff = TRUEdiff = FALSE
```

```
keep.source = TRUEsourceKeepSource = TRUEDESCRIPTIONlines
```

```
lines = "show"
```

```
lines = "both"
```

```
RShowDoc("R-exts")
```

```
Rprof
```

```
tracememduplicate
```

```
Rprofmem
```

```
https://developer.r-project.org/memory-profiling.html
```

```
## Not run:
```

```
## Rprof() is not available on all platforms
```

```
Rprof(tmp <- tempfile())
```

```
example(glm)
```

```
Rprof()
```

```
summaryRprof(tmp)
```

```
unlink(tmp)
```

```
## End(Not run)
```

Sweave

Sweave

```
Sweave(file, driver = RweaveLatex(),  
       syntax = getOption("SweaveSyntax"), encoding = "", ...)
```

```
Stangle(file, driver = Rtangle(),  
        syntax = getOption("SweaveSyntax"), encoding = "", ...)
```

file	file
driver	list vignette("Sweave")
syntax	NULL "SweaveSyntax"
encoding	file
...	RweaveLatexSetup RtangleSetup

```
SweaveStanglesource\Sexpr{ }Stangle
```

```
StangleSweave
```

```
SWEAVE_OPTIONSkey=value\SweaveOpts
```

```
encoding
```

```
\usepackage[foo]{inputenc}
```

```
foolatin1latin2utf8cp1252cp1250
```

```
%\SweaveUTF8
```

```
encoding = "bytes"
```

```
syntax      =      NULLextensionSweaveSyntaxNowebextension      =      "[.][rsRS]nw$"  
SweaveSyntaxLatexextension = "[.][rsRS]tex$"SweaveSyntax.*
```

RweaveLatexRtangle

tools::buildVignette

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")
```

```
## enforce par(ask = FALSE)
options(device.ask.default = FALSE)
```

```
## create a LaTeX file - in the current working directory, getwd():
Sweave(testfile)
```

```
## This can be compiled to PDF by
## tools::texi2pdf("Sweave-test-1.tex")
```

```
## or outside R by
##
## R CMD texi2pdf Sweave-test-1.tex
## on Unix-alikes which sets the appropriate TEXINPUTS path.
##
## On Windows,
## Rcmd texify --pdf Sweave-test-1.tex
## if MiKTeX is available.
```

```
## create an R source file from the code chunks
Stangle(testfile)
## which can be sourced, e.g.
source("Sweave-test-1.R")
```

SweaveSyntConv

Sweave

```
SweaveSyntConv(file, syntax, output = NULL)
```

```
file
syntax      SweaveSyntax
output      file
```


RweaveLatexRtangle

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")

## convert the file to latex syntax
SweaveSyntConv(testfile, SweaveSyntaxLatex)

## and run it through Sweave
Sweave("Sweave-test-1.Stex")
```

tar

```
tar(tarfile, files = NULL,
    compression = c("none", "gzip", "bzip2", "xz", "zstd"),
    compression_level = 6, tar = Sys.getenv("tar"),
    extra_flags = "")
```

```
tarfile      path.expand
files
compression
compression_level
             gzfile
tar           shQuote tar
extra_flags   tar
```

```
tar tarfile tar "internal" "" TAR
extra_flag star -h -L --acls --exclude-backups --exclude-vcs --force-local
tar TAR_OPTIONS --force-local TAR tar tar bsdtar --force-local TAR_OPTIONS
tar --format=ustar tar --help --format=gnu --format=posix pax tar bsdtar --format=ustar
tar tar bsdtar
files = '.' tar tarfile
```

system0

pax
tarbsdtarbsdtarlibarchivetargnutargtarconfigregnutargtartar

tartartarpaxtar**untar**

tar
bsdtar

tarnul

tartar-z-j-J--zstdgzipbzip2xzzstdtarextra_flags = "-I lz4""--lzip""--lzop"
extra_flagsbsdtar--lz4--lzop--lrzip--use-compress-program lz4extra_flags
--xz-Jextra_flags = "--xz"compression = "xz"xzzstd
tarzstdzstd
tarxzzstdbsdtarxzzstd
tarzstd -T0

tar._**untar**
tartarCOPYFILE_DISABLE=1tar

[https://en.wikipedia.org/wiki/Tar_\(file_format\)](https://en.wikipedia.org/wiki/Tar_(file_format))https://pubs.opengroup.org/onlinepubs/9699919799/utilities/pax.html#tag_20_92_13_06paxtar
<https://github.com/libarchive/libarchive/wiki/FormatTar>
untar

toLatex

```
toBibtex(object, ...)
toLatex(object, ...)
## S3 method for class 'Bibtex'
print(x, prefix = "", ...)
## S3 method for class 'Latex'
print(x, prefix = "", ...)
```

```

object      toBibtexToLatex
x           "Bibtex""Latex"
prefix
...         writeLines

```

```
"Bibtex""Latex"
```

```
citEntrySessionInfo
```

```
txtProgressBar
```

```

txtProgressBar(min = 0, max = 1, initial = 0, char = "=",
               width = NA, title, label, style = 1, file = "")

```

```

getTxtProgressBar(pb)
setTxtProgressBar(pb, value, title = NULL, label = NULL)
## S3 method for class 'txtProgressBar'
close(con, ...)

```

```

minmax      min < max
initialvalue
char
width       charNAgetOption("width")
style
file        ""stderr()
pbcon       "txtProgressBar"
titlelabel
...

```

```

txtProgressBar
setTxtProgessBarNAvalueinitial
close
style = 1style = 2charstyle = 2style = 3|

```

```
txtProgressBar"txtProgressBar"
getTxtProgressBarsetTxtProgressBarsetTxtProgressBar
```

```
stylestyle = 1\rstdout()
```

```
winProgressBartkProgressBar
```

```
# slow
testit <- function(x = sort(runif(20)), ...)
{
  pb <- txtProgressBar(...)
  for(i in c(0, x, 1)) {Sys.sleep(0.5); setTxtProgressBar(pb, i)}
  Sys.sleep(1)
  close(pb)
}
testit()
testit(runif(10))
testit(style = 3)
testit(char=' \u27a4')
```

```
type.convert
```

```
type.convert(x, ...)
## Default S3 method:
type.convert(x, na.strings = "NA", as.is, dec = ".",
             numerals = c("allow.loss", "warn.loss", "no.loss"),
             tryLogical = TRUE, ...)
## S3 method for class 'data.frame'
type.convert(x, ...)
## S3 method for class 'list'
type.convert(x, ...)

x
na.strings      NA
as.is           character
dec
numerals        xdouble
                numerals = "allow.loss"
                numerals = "warn.loss" warningnumerals = "allow.loss"
                numerals = "no.loss" xfactorcharacteras.is
tryLogical      logicalFTFALSETRUEena.stringslogical
...
```

```

read.tablex
as.is = FALSEfactor
NA
tryLogicalFTFALSETRUEna.stringscharacter"F"tryLogical = FALSEna.stringsNaNInf
infinityna.stringsas.is = FALSE
as.is

```

```

x

```

```

read.tableclassstorage.mode

```

```

## Numeric to integer
class(rivers)
x <- type.convert(rivers, as.is = TRUE)
class(x)

## Convert many columns
auto <- type.convert(mtcars, as.is = TRUE)
str(mtcars)
str(auto)

## Convert matrix
phones <- type.convert(WorldPhones, as.is = TRUE)
storage.mode(WorldPhones)
storage.mode(phones)

## Factor or character
chr <- c("A", "B", "B", "A")
ch2 <- c("F", "F", "NA", "F")
(fac <- factor(chr))
type.convert(chr, as.is = FALSE) # -> factor
type.convert(fac, as.is = FALSE) # -> factor
type.convert(chr, as.is = TRUE) # -> character
type.convert(fac, as.is = TRUE) # -> character
type.convert(ch2, as.is = TRUE) #-> logical
type.convert(ch2, as.is = TRUE, tryLogical=FALSE) #-> character

```

```

untar

```

```

untar(tarfile, files = NULL, list = FALSE, exdir = ".",
      compressed = NA, extras = NULL, verbose = FALSE,
      restore_times = TRUE,
      support_old_tars = Sys.getenv("R_SUPPORT_OLD_TARS", FALSE),
      tar = Sys.getenv("TAR"))

```

```

tarfile      path.expandtarfilegzfile(.)gzcon(.)"gzip"gzfile()tar()
files
list         TRUEtar -tftar -xf
exdir        tar -C
compressed    tar"gzip""bzip2""xz""zstd"TRUEgzipFALSENA

```

```

extras       NULL-p-Ptar
verbose      tar
restore_times -m

```

```

support_old_tars
              tarcompressedtar
              tarcompressed = NA
tar          "internal""tar

```

```

tartarfiletar"internal""tar.exe
gzipbzip2xz
tar"internal"

```

```

  tarxzzstd
    configureTARgtargnutar
bsdtar tarzstdzstd
  xzzstd
  bsdtartar.exe
  tarxzzxsupport_old_tars = TRUE
  R_GZIPCMDgzipR_BZIPCMDbzip2xzzstd

```

```

compressedextrasverbosetar
tarlrziplzmalz4lzopgzipbzip2xzzstdtartarextras = "-I lz4"bsdtarextras =
"--use-compress-program lz4".tar.Zcompress
gzfilecompressgzipbzip2xzzstdlzmapax

```

```

      unsupported entry type 'x'
"A-Z"tar
      using pax extended headers

```

```

untar
tar._

```

```
list = TRUE
systemtar0L
```

```
tarunzip
```

```
unzip
```

```
unzip(zipfile, files = NULL, list = FALSE, overwrite = TRUE,
      junkpaths = FALSE, exdir = ".", unzip = "internal",
      setTimes = FALSE)
```

zipfile	<code>path.expand</code>
files	
list	<code>TRUEunzip -l</code>
overwrite	<code>TRUEunzip -ounzip -n</code>
junkpaths	<code>TRUEunzip -j</code>
exdir	<code>unzip -d</code>
unzip	<code>getOption("unzip")unzip</code>
setTimes	

```
list = TRUENameLengthDate"POSIXct"
"internal"
```

```
zip 3.0unzip = "unzip"unzip 6.00bzip2unzip
unziplist = TRUEUNAunzip 6.00
```

```
unzip 6.00"../"
```

```
zlibminizipzlibhttps://zlib.net/
```

```
unz
zipunzip()untartar
```

update.packages

old.packagesupdate.packages
new.packages

```
update.packages(lib.loc = NULL, repos = getOption("repos"),
               contriburl = contrib.url(repos, type),
               method, instlib = NULL,
               ask = TRUE, available = NULL,
               oldPkgs = NULL, ..., checkBuilt = FALSE,
               type = getOption("pkgType"))

old.packages(lib.loc = NULL, repos = getOption("repos"),
             contriburl = contrib.url(repos, type),
             instPkgs = installed.packages(lib.loc = lib.loc, ...),
             method, available = NULL, checkBuilt = FALSE, ...,
             type = getOption("pkgType"))

new.packages(lib.loc = NULL, repos = getOption("repos"),
             contriburl = contrib.url(repos, type),
             instPkgs = installed.packages(lib.loc = lib.loc, ...),
             method, available = NULL, ask = FALSE, ...,
             type = getOption("pkgType"))
```

lib.loc	NULL.libPaths
repos	"https://cloud.r-project.org"
contriburl	repostype = "both"
method	download.fileold.packagesNULLavailable
instlib	
ask	"graphics"select.listask = TRUEasktype = "both" "install.packages.compile.from.source"
available	available.packagesNULLavailable.packagestype = "both"
checkBuilt	TRUE3.4
oldPkgs	update.packages()old.packages
instPkgs	installed.packages(lib.loc = lib.loc)installed.packages()
...	destdirdependenciesinstall.packagesignore_repo_cache max_repo_cache_agenoCacheavailable.packagesinstalled.packages
type	install.packages

```
old.packagesavailable.packagesinstPkgsinstalled.packagescheckBuilt = TRUE
new.packagesask != FALSElib.loc
update.packageslib.locask = TRUEinstlib
available.packagesavailable = NULLavailable.packages(contriburl = contriburl,
method = method)
```



```
update.packagesNULL
old.packagesNULL"Package""LibPath""Installed""Built""ReposVer""Repository"
new.packagesask
```

```
dependenciesinstall.packagesupdate.packages
```

```
install.packagesavailable.packagesdownload.packagesinstalled.packages
contrib.url
install.packagesoptions
download.file
INSTALLREMOVEremove.packageslibrary.packagesread.dcf
```

upgrade

```
upgrade(object, ...)
```

```
object
...
```

```
"packageStatus"
```

url.show

```
file.show
```

```
url.show(url, title = url, file = tempfile(),
         delete.file = TRUE, method, ...)
```

```
url
title
file
delete.file
method      download.file
...         file.show
```

[urlfile.showdownload.file](#)

```
## Not run: url.show("https://www.stats.ox.ac.uk/pub/datasets/csb/ch3a.txt")
```

URLencode

```
URLencode(URL, reserved = FALSE, repeated = FALSE)
URLdecode(URL)
```

URL
reserved
repeated

- _ . ~%
! \$ & ' () * + , ; = : / ? @ # []URLencodefile://http://
%xx

<https://www.rfc-editor.org/info/std66>

```
(y <- URLencode("a url with spaces and / and @"))
URLdecode(y)
(y <- URLencode("a url with spaces and / and @", reserved = TRUE))
URLdecode(y)

URLdecode(z <- "ab%20cd")
c(URLencode(z), URLencode(z, repeated = TRUE)) # first is usually wanted

## both functions support character vectors of length > 1
y <- URLdecode(URLencode(c("url with space", "another one")))
```

utils-deprecated

[DeprecatedDefunct](#)

View

View(x, title)

x
title xData:

[xformat.data.framedata.entry](#)
1:nrowrow.names

[Rconsole](#)

NULL

[edit.data.framedata.entry](#)

vignette

```
vignette(topic, package = NULL, lib.loc = NULL, all = TRUE)
```

```
## S3 method for class 'vignette'  
print(x, ...)  
## S3 method for class 'vignette'  
edit(name, ...)
```

```
topic      package::topic  
package    NULLall  
lib.loc    NULLNULL  
all        TRUElib.locFALSE  
xname      vignette  
...        printfile.editedit
```

```
vignette  
pdfviewer  
topic  
"packageIQR"
```

```
browseVignettesRShowDoc("", package = "")
```

```
## List vignettes from all *attached* packages  
vignette(all = FALSE)
```

```
## List vignettes from all *installed* packages (can take a long time!)  
vignette(all = TRUE)
```

```
## List all vignettes of a specific package (package 'grid' has several)  
vignette(package = "grid")
```

```
## Open one of the 'grid' vignettes  
if(interactive()) {  
  ## vignette("rotated", package = "grid")  
  ## Or, the same:  
  vignette(grid::rotated) # calling the print() method  
}
```

```
## Now open the 'grid' intro vignette -- without specifying the package
```

```

## Not run: vignette("grid")
## OK, but warns as this topic is ambiguous: both {grid} and {lattice} have it.
## => Specify the 'package' argument or use the <package>::<topic> syntax
## (also accelerates vignette retrieval, esp. with many installed packages):
v1 <- vignette("grid", package = "grid") # not "printed" (opened) yet
if(inherits(v1, "vignette")) { # it was found installed
  ## Not run: print(v1) # open it
  str(v1)
  ## To open the associated R code in an editor:
  ## Not run: edit(v1) # e.g., to send lines ...
}

```

warnErrList

```

"error"tryCatchxwarning"warningMsg"

```

```

warnErrList(x, warn = TRUE, errValue = NULL)

```

```

x          listtryCatch(*, error = identity)
warn       warning()
errValue

```

```

listxerrValueNULL"warningMsg"

```

```

warnErrList()lmList()nlsList()

```

```

## Regression for each Chick:
ChWtgrps <- split(ChickWeight, ChickWeight[, "Chick"])
sapply(ChWtgrps, nrow)# typically 12 obs.
nlis1 <- lapply(ChWtgrps, function(DAT) tryCatch(error = identity,
  lm(weight ~ (Time + I(Time^2)) * Diet, data = DAT)))
nl1 <- warnErrList(nlis1) #-> warning :
## 50 times the same error (as Diet has only one level in each group)
stopifnot(sapply(nl1, is.null)) ## all errors --> all replaced by NULL
nlis2 <- lapply(ChWtgrps, function(DAT) tryCatch(error = identity,
  lm(weight ~ Time + I(Time^2), data = DAT)))
nl2 <- warnErrList(nlis2)
stopifnot(identical(nl2, nlis2)) # because there was *no* error at all
nlis3 <- lapply(ChWtgrps, function(DAT) tryCatch(error = identity,
  lm(weight ~ poly(Time, 3), data = DAT)))
nl3 <- warnErrList(nlis3) # 1 error caught:
stopifnot(inherits(nlis3[[1]], "error")
  , identical(nl3[-1], nlis3[-1])
  , is.null(nlis3[[1]]))
)

```

```
## With different error messages
if(requireNamespace("nlme")) { # almost always, as it is recommended
  data(Soybean, package="nlme")
  attr(Soybean, "formula") #-> weight ~ Time | Plot => split by "Plot":
  L <- lapply(split(Soybean, Soybean[, "Plot"]),
              function(DD) tryCatch(error = identity,
                                    nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = DD)))
  Lw <- warnErrList(L)
} # if <nlme>
```

winDialog

```
winDialog(type = c("ok", "okcancel", "yesno", "yesnocancel"),
          message)
```

```
winDialogString(message, default)
```

```
type
```

```
message
```

```
default
```

```
winDialogNULL
```

```
winDialogStringOkNULLCancel
```

```
ReturnEscYN
```

```
winMenuAdd
```

```
file.choose
```

```
windlgsGraphApp
```

```
## Not run: winDialog("yesno", "Is it OK to delete file blah")
```

winextras

win.version()

win.version[sessionInfo](#)[bug.report](#)

GetVersionEx

```
if(.Platform$OS.type == "windows")  
  print(win.version())
```

winMenus

Rgui

```
winMenuAdd(menuname)  
winMenuAddItem(menuname, itemname, action)  
winMenuDel(menuname)  
winMenuDelItem(menuname, itemname)  
winMenuNames()  
winMenuItems(menuname)
```

menuname

itemname

action "enable""disable"

menunamewinMenuAddItem

\$ConsoleMain

\$ConsolePopup

\$Graph<n>Main <n>

\$Graph<n>Popup <n>

winMenuAddItem

menuname

action"none"

winMenuNames

winMenuItems

winMenuDelwinMenuDelItem

NULL

RguiESSRStudio

winDialog

Not run:

winMenuAdd("Testit")

winMenuAddItem("Testit", "one", "aaaa")

winMenuAddItem("Testit", "two", "bbbb")

winMenuAdd("Testit/extras")

winMenuAddItem("Testit", "-", "")

winMenuAddItem("Testit", "two", "disable")

winMenuAddItem("Testit", "three", "cccc")

winMenuAddItem("Testit/extras", "one more", "ddd")

winMenuAddItem("Testit/extras", "and another", "eee")

winMenuAdd("\$ConsolePopup/Testit")

winMenuAddItem("\$ConsolePopup/Testit", "six", "fff")

winMenuNames()

winMenuItems("Testit")

End(Not run)

winProgressBar

```
winProgressBar(title = "R progress bar", label = "",
               min = 0, max = 1, initial = 0, width = 300)

getWinProgressBar(pb)
setWinProgressBar(pb, value, title = NULL, label = NULL)
## S3 method for class 'winProgressBar'
close(con, ...)
```

```
titlelabel
minmax
initialvalue
width
pbcon      "winProgressBar"
...
```

```
winProgressBar
setWinProgressBarNULLNAvalue
close
```

```
winProgressBar"winProgressBar"
getWinProgressBarsetWinProgressBarsetWinProgressBar
```

[txtProgressBar](#)[tkProgressBar](#)

`write.table`

`write.tablex`

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
```

```
write.csv(...)
write.csv2(...)
```

x	x
file	""
append	fileTRUEFALSE
quote	TRUEFALSETRUEFALSE
sep	x
eol	eol = "\r\n"eol = "\r"
na	
dec	
row.names	xx
col.names	xxcol.names = NA
qmethod	"escape"write.table"double"write.csvwrite.csv2
fileEncoding	file
...	write.tableappendcol.namessepdecqmethod

```
row.names = TRUE
```

```
as.matrixcol.namesquote
as.character
quote
decI()options("OutDec")
"scipen"optionsdigits = 15formatwrite.table

file
file = file("filename", "wb")
```

```

write.table(col.names = NA, row.names = TRUE

read.csv(file = "<filename>", row.names = 1)

write.csv(write.csv2sepdecqmethod = "double", col.names = NA, row.names = TRUE, TRUE
write.csv".")
write.csv2
append.col.names = sepdecqmethod
fileEncoding = "UTF-16LE" "CP1252"
https://www.rfc-editor.org/rfc/rfc4180 write.csv(eol = "\r\n"

write.table

read.table write
write.matrix

x <- data.frame(a = "a \" quote", b = pi)
tf <- tempfile(fileext = ".csv")

## To write a CSV file for input to Excel one might use
write.table(x, file = tf, sep = ",", col.names = NA,
            qmethod = "double")
file.show(tf)
## and to read this file back into R one needs
read.table(tf, header = TRUE, sep = ",", row.names = 1)
## NB: you do need to specify a separator if qmethod = "double".

### Alternatively
write.csv(x, file = tf)
read.csv(tf, row.names = 1)
## or without row names
write.csv(x, file = tf, row.names = FALSE)
read.csv(tf)

## Not run:
## To write a file in Mac Roman for simple use in Mac Excel 2004/8
write.csv(x, file = "foo.csv", fileEncoding = "macroman")
## or for Windows Excel 2007/10
write.csv(x, file = "foo.csv", fileEncoding = "UTF-16LE")

## End(Not run)

```

zip

zip

```
zip(zipfile, files, flags = "-r9X", extras = "",  
    zip = Sys.getenv("R_ZIPCMD", "zip"))
```

zipfile [path.expand](#)

files

flags

extras

zip

zipR_ZIPCMDetc/Renvironzipzip

flagszip

extras-x-iextrassystemshQuote

[unzipunztaruntar](#)

KernSmooth

bkde

```
bkde(x, kernel = "normal", canonical = FALSE, bandwidth,
      gridsize = 401L, range.x, truncate = TRUE)
```

x	
bandwidth	bandwidthbandwidthx
kernel	kernel"normal""box""epanech""biweight""triweight"
canonical	TRUE
gridsize	
range.x	x
truncate	TRUExrange.x

xy

x	x
y	x

densitydpikhistksmooth

```
data(geyser, package="MASS")
x <- geyser$duration
est <- bkde(x, bandwidth=0.25)
plot(est, type="l")
```

bkde2D

```
bkde2D(x, bandwidth, gridsize = c(51L, 51L), range.x, truncate = TRUE)
```

```
x
bandwidth
gridsize
range.x      x
truncate     xrange.x
```

```
x1
x2
fhat      x1x2
```

```
x1x2fhat
```

bkdedensityhist

```
data(geyser, package="MASS")
x <- cbind(geyser$duration, geyser$waiting)
est <- bkde2D(x, bandwidth=c(0.7, 7))
contour(est$x1, est$x2, est$fhat)
persp(est$fhat)
```

bkfe

```
bkfe(x, drv, bandwidth, gridsize = 401L, range.x, binned = FALSE,  
      truncate = TRUE)
```

x

drv

bandwidth

gridsize

range.x x

binned TRUExy

truncate TRUExrange.x

drvdrv

```
data(geyser, package="MASS")  
x <- geyser$duration  
est <- bkfe(x, drv=4, bandwidth=0.3)
```

dpih

```
dpih(x, scalest = "minim", level = 2L, gridsize = 401L,  
     range.x = range(x), truncate = TRUE)
```

```
x  
scalest  
      "stdev"  
      "iqr"  
      "minim" "stdev" "iqr"  
level  
gridsize  
range.x  
truncate      truncateTRUE range.x
```

hist

```
data(geyser, package="MASS")  
x <- geyser$duration  
h <- dpih(x)  
bins <- seq(min(x)-h, max(x)+h, by=h)  
hist(x, breaks=bins)
```

dpik

```
dpik(x, scalest = "minim", level = 2L, kernel = "normal",  
      canonical = FALSE, gridsize = 401L, range.x = range(x),  
      truncate = TRUE)
```

```
x  
scalest  
      "stdev"  
      "iqr"  
      "minim" "stdev" "iqr"  
level  
kernel      kernel "normal" "box" "epanech" "biweight" "triweight"  
canonical    TRUE  
gridsize  
range.x      x  
truncate    TRUExrange.x
```

[bkdedensitysmooth](#)

```
data(geyser, package="MASS")  
x <- geyser$duration  
h <- dpik(x)  
est <- bkde(x, bandwidth=h)  
plot(est, type="l")
```

dpill

```
dpill(x, y, blockmax = 5, divisor = 20, trim = 0.01, proptrun = 0.05,  
      gridsize = 401L, range.x, truncate = TRUE)
```

x	
y	x
blockmax	
divisor	
trim	x
proptrun	x
gridsize	
range.x	x
truncate	TRUExrange.x

C_p

xy

[ksmoothlocpoly](#)

```
data(geyser, package = "MASS")  
x <- geyser$duration  
y <- geyser$waiting  
plot(x, y)  
h <- dpill(x, y)  
fit <- locpoly(x, y, bandwidth = h)  
lines(fit)
```

locpoly

```
locpoly(x, y, drv = 0L, degree, kernel = "normal",
        bandwidth, gridsize = 401L, bwdisc = 25,
        range.x, binned = FALSE, truncate = TRUE)
```

x	
bandwidth	gridsize
y	x
drv	
degree	drvdegreedrv
kernel	"normal"
gridsize	
bwdisc	bandwidth
range.x	x
binned	TRUExy
truncate	TRUExrange.x

yyx

x	
y	x

gridsize

[bkdedensitydpillksmoothloesssmoothsupsmu](#)

```
data(geyser, package = "MASS")
# local linear density estimate
x <- geyser$duration
est <- locpoly(x, bandwidth = 0.25)
plot(est, type = "l")

# local linear regression estimate
y <- geyser$waiting
plot(x, y)
fit <- locpoly(x, y, bandwidth = 0.25)
lines(fit)
```


MASS

abbey

abbey

accdeaths

accdeaths

7798 7406 8363 8460 9217 9316

addterm

lmglm

addterm(object, ...)

Default S3 method:

```
addterm(object, scope, scale = 0, test = c("none", "Chisq"),
        k = 2, sorted = FALSE, trace = FALSE, ...)
```

S3 method for class 'lm'

```
addterm(object, scope, scale = 0, test = c("none", "Chisq", "F"),
        k = 2, sorted = FALSE, ...)
```

S3 method for class 'glm'

```
addterm(object, scope, scale = 0, test = c("none", "Chisq", "F"),
        k = 2, sorted = FALSE, trace = FALSE, ...)
```

object

scope

scale lmaovglmscale

test lmaovglm1m

k k=2k = log(n)

sorted

trace TRUE

...

lm

"anova"

[droptermstepAIC](#)

```
quine.hi <- aov(log(Days + 2.5) ~ .^4, quine)
```

```
quine.lo <- aov(log(Days+2.5) ~ 1, quine)
```

```
addterm(quine.lo, quine.hi, test="F")
```

```
house.glm0 <- glm(Freq ~ Infl*Type*Cont + Sat, family=poisson,
                  data=housing)
```

```
addterm(house.glm0, ~. + Sat:(Infl+Type+Cont), test="Chisq")
```

```
house.glm1 <- update(house.glm0, . ~ . + Sat*(Infl+Type+Cont))
```

```
addterm(house.glm1, ~. + Sat:(Infl+Type+Cont)^2, test = "Chisq")
```

Aids2

Aids2

state "NSW ""other"
sex
diag
death
status "A""D"
T.categ
age

Animals

Animals

body
brain

Animalsanimals

anorexia

anorexia

anorexia

Treat "Cont" "CBT" "FT"

Prewt

Postwt

anova.negbin

```
## S3 method for class 'negbin'
anova(object, ..., test = "Chisq")
```

```
object      "negbin" "glm" "lm" glm.nb\(\)
...         "negbin"
test        testanova.glm
```

```
anova() "negbin" anova(x) xanova.negbin(x)
```

```
theta "negbin" theta
```

[glm.negative.binomial.summary.negbin](#)

```
m1 <- glm.nb(Days ~ Eth*Age*Lrn*Sex, quine, link = log)
m2 <- update(m1, . ~ . - Eth:Age:Lrn:Sex)
anova(m2, m1)
anova(m2)
```

area

```
area(f, a, b, ..., fa = f(a, ...), fb = f(b, ...),  
      limit = 10, eps = 1e-05)
```

```
f          S  
a  
b  
...  
fa  
fb  
limit  
eps
```

```
abf(x)
```

```
area(sin, 0, pi) # integrate the sin function from 0 to pi.
```

bacteria

```
bacteria
```

```
ny  
ap  
hilo
```

```
placebodrugdrug+aphilo
```

https://www.menzies.edu.au/icms_docs/172302_2000_Annual_report.pdf

```
contrasts(bacteria$trt) <- structure(contr.sdif(3),
  dimnames = list(NULL, c("drug", "encourage")))
## fixed effects analyses
summary(glm(y ~ trt * week, binomial, data = bacteria))
summary(glm(y ~ trt + week, binomial, data = bacteria))
summary(glm(y ~ trt + I(week > 2), binomial, data = bacteria))

# conditional random-effects analysis
library(survival)
bacteria$Time <- rep(1, nrow(bacteria))
coxph(Surv(Time, unclass(y)) ~ week + strata(ID),
  data = bacteria, method = "exact")
coxph(Surv(Time, unclass(y)) ~ factor(week) + strata(ID),
  data = bacteria, method = "exact")
coxph(Surv(Time, unclass(y)) ~ I(week > 2) + strata(ID),
  data = bacteria, method = "exact")

# PQL glmm analysis
library(nlme)
## IGNORE_RDIFF_BEGIN
summary(glmmPQL(y ~ trt + I(week > 2), random = ~ 1 | ID,
  family = binomial, data = bacteria))
## IGNORE_RDIFF_END
```

bandwidth.nrd

bandwidth.nrd(x)

x

widthdensity

```
# The function is currently defined as
function(x)
{
  r <- quantile(x, c(0.25, 0.75))
  h <- (r[2] - r[1])/1.34
  4 * 1.06 * min(sqrt(var(x)), h) * length(x)^(-1/5)
}
```

bcv

```
bcv(x, nb = 1000, lower, upper)
```

```
x
```

```
nb
```

```
lowerupper
```

```
ucvwidth.SJdensity
```

```
bcv(geyser$duration)
```

beav1

beav1

beav1

day

time 0330

temp

activ

beav2

```
beav1 <- within(beav1,
  hours <- 24*(day-346) + trunc(time/100) + (time%%100)/60)
plot(beav1$hours, beav1$temp, type="l", xlab="time",
  ylab="temperature", main="Beaver 1")
usr <- par("usr"); usr[3:4] <- c(-0.2, 8); par(usr=usr)
lines(beav1$hours, beav1$activ, type="s", lty=2)
temp <- ts(c(beav1$temp[1:82], NA, beav1$temp[83:114]),
  start = 9.5, frequency = 6)
activ <- ts(c(beav1$activ[1:82], NA, beav1$activ[83:114]),
  start = 9.5, frequency = 6)

acf(temp[1:53])
acf(temp[1:53], type = "partial")
ar(temp[1:53])
act <- c(rep(0, 10), activ)
X <- cbind(1, act = act[11:125], act1 = act[10:124],
  act2 = act[9:123], act3 = act[8:122])
alpha <- 0.80
stemp <- as.vector(temp - alpha*lag(temp, -1))
sX <- X[-1, ] - alpha * X[-115,]
beav1.ls <- lm(stemp ~ -1 + sX, na.action = na.omit)
summary(beav1.ls, correlation = FALSE)
rm(temp, activ)
```

beav2

beav2

beav2

day

time 0330

temp

activ

beav1

```
attach(beav2)
beav2$hours <- 24*(day-307) + trunc(time/100) + (time%%100)/60
plot(beav2$hours, beav2$temp, type = "l", xlab = "time",
     ylab = "temperature", main = "Beaver 2")
usr <- par("usr"); usr[3:4] <- c(-0.2, 8); par(usr = usr)
lines(beav2$hours, beav2$activ, type = "s", lty = 2)

temp <- ts(temp, start = 8+2/3, frequency = 6)
activ <- ts(activ, start = 8+2/3, frequency = 6)
acf(temp[activ == 0]); acf(temp[activ == 1]) # also look at PACFs
ar(temp[activ == 0]); ar(temp[activ == 1])

arima(temp, order = c(1,0,0), xreg = activ)
dreg <- cbind(sin = sin(2*pi*beav2$hours/24), cos = cos(2*pi*beav2$hours/24))
arima(temp, order = c(1,0,0), xreg = cbind(active=activ, dreg))

## IGNORE_RDIFF_BEGIN
library(nlme) # for gls and corAR1
beav2.gls <- gls(temp ~ activ, data = beav2, correlation = corAR1(0.8),
                 method = "ML")
summary(beav2.gls)
summary(update(beav2.gls, subset = 6:100))
detach("beav2"); rm(temp, activ)
## IGNORE_RDIFF_END
```

Belgian-phones

year
calls

phones

biopsy

biopsy

ID
V1
V2
V3
V4
V5
V6
V7
V8
V9
class "benign""malignant"

birthwt

birthwt

birthwt

low

age

lwt

race 123

smoke

ptl

ht

ui

ftv

bwt

```
bwt <- with(birthwt, {  
  race <- factor(race, labels = c("white", "black", "other"))  
  ptd <- factor(ptl > 0)  
  ftv <- factor(ftv)  
  levels(ftv)[-1:2] <- "2+"  
  data.frame(low = factor(low), age, lwt, race, smoke = (smoke > 0),  
             ptd, ht = (ht > 0), ui = (ui > 0), ftv)  
})  
options(contrasts = c("contr.treatment", "contr.poly"))  
glm(low ~ ., binomial, bwt)
```

Boston

Boston

Boston

crim

zn

indus

chas

nox

rm

age

dis

rad

tax

ptratio

black $1000(Bk - 0.63)^2 Bk$

lstat

medv

boxcox

```

boxcox(object, ...)

## Default S3 method:
boxcox(object, lambda = seq(-2, 2, 1/10), plotit = TRUE,
        interp, eps = 1/50, xlab = expression(lambda),
        ylab = "log-Likelihood", ...)

## S3 method for class 'formula'
boxcox(object, lambda = seq(-2, 2, 1/10), plotit = TRUE,
        interp, eps = 1/50, xlab = expression(lambda),
        ylab = "log-Likelihood", ...)

## S3 method for class 'lm'
boxcox(object, lambda = seq(-2, 2, 1/10), plotit = TRUE,
        interp, eps = 1/50, xlab = expression(lambda),
        ylab = "log-Likelihood", ...)

```

```

object      lmaov
lambda      lambda(-2,2)
plotit
interp      TRUElambda
eps         lambda = 0
xlab        "lambda"
ylab        "log-Likelihood"
...

```

```

lambda

```

```

plotit = TRUElambda
lambda
lambda
interp = TRUE

```

```

boxcox(Volume ~ log(Height) + log(Girth), data = trees,
        lambda = seq(-0.25, 0.25, length.out = 10))

boxcox(Days+1 ~ Eth*Sex*Age*Lrn, data = quine,
        lambda = seq(-0.05, 0.45, length.out = 20))

```

cabbages

cabbages

cabbages

Cult c39c52
Date d16d20d21
HeadWt
VitC

caith

caith

```
## IGNORE_RDIFF_BEGIN
## The signs can vary by platform
corresp(caith)
## IGNORE_RDIFF_END
dimnames(caith)[[2]] <- c("F", "R", "M", "D", "B")
par(mfcol=c(1,3))
plot(corresp(caith, nf=2)); title("symmetric")
plot(corresp(caith, nf=2), type="rows"); title("rows")
plot(corresp(caith, nf=2), type="col"); title("columns")
par(mfrow=c(1,1))
```

Cars93

Cars93

Cars93

Manufacturer

Model

Type "Small""Sporty""Compact""Midsize""Large""Van"

Min.Price

Price Min.PriceMax.Price

Max.Price

MPG.city

MPG.highway

AirBags

DriveTrain

Cylinders

EngineSize

Horsepower

RPM

Rev.per.mile

Man.trans.avail

Fuel.tank.capacity

Passengers

Length

Wheelbase

Width

Turn.circle

Rear.seat.room

Luggage.room

Weight

Origin

Make

cats

cats

Sex "F" "M"

Bwt

Hwt

cement

cement

x1, x2, x3, x4

y

lm(y ~ x1 + x2 + x3 + x4, cement)

chem

chem

con2tr

con2tr(obj)

obj

xyzcontour

con2trxyz

confint-MASS

glmnl

polr

`contr.sdif`

`contr.sdif(n, contrasts = TRUE, sparse = FALSE)`

`n`
`contrasts` `n - 1n`
`sparse` `contrasts = FALSE`

`contrastsTRUE``Enn - 1nncontrastsFALSE`

[contr.treatment](#)[contr.sum](#)[contr.helmert](#)

`(A <- contr.sdif(6))`
`zapsmall(ginv(A))`

`coop`

`coop`

Lab L1L2L6
Spc S1S2S7
Bat B1B2B3Spc/Lab
Conc *g/kg*

[chemabbey](#)

corresp

```
corresp(x, ...)  
  
## S3 method for class 'matrix'  
corresp(x, nf = 1, ...)  
  
## S3 method for class 'factor'  
corresp(x, y, ...)  
  
## S3 method for class 'data.frame'  
corresp(x, ...)  
  
## S3 method for class 'xtabs'  
corresp(x, ...)  
  
## S3 method for class 'formula'  
corresp(formula, data, ...)  
  
xformula      "xtabs" ~ F1 + F2F1F2  
nf  
y  
data  
...  
  
plotnf=1nfbiplotA = Dr^(-1/2) U LB = Dc^(-1/2) V LU L V  
  
"correspondence"printplotbiplot  
  
svdprincomp  
  
## IGNORE_RDIFF_BEGIN  
## The signs can vary by platform  
(ct <- corresp(~ Age + Eth, data = quine))  
plot(ct)  
  
corresp(caith)  
biplot(corresp(caith, nf = 2))  
## IGNORE_RDIFF_END
```

cov.rob

goodcov.mvecov.mcd

```
cov.rob(x, cor = FALSE, quantile.used = floor((n + p + 1)/2),  
        method = c("mve", "mcd", "classical"),  
        nsamp = "best", seed)
```

```
cov.mve(...)  
cov.mcd(...)
```

```
x  
cor  
quantile.used  good  
method         cov.mvecov.mcdmvecmcd  
nsamp          "best""exact""sample""sample"min(5*p, 3000)"best""exact"  
seed           RNGkind.Random.seed  
...           cov.robmethod
```

"mve"quantile.used"mcd"goodgood

"mve"p"mcd"quantile.used
 $2^{31} - 1$ combn(NROW(x), NCOL(x) + 1)

```
center  
cov  
cor          cor = TRUE  
sing  
crit  
best         quantile.used  
n.obs
```

[lqs](#)

```
set.seed(123)
cov.rob(stackloss)
cov.rob(stack.x, method = "mcd", nsamp = "exact")
```

`cov.trob`

```
cov.trob(x, wt = rep(1, n), cor = FALSE, center = TRUE, nu = 5,
         maxit = 25, tol = 0.01)
```

```
x
wt          iwt[i]
cor          cor = TRUEcor = FALSE
center       center = FALSEcenter = TRUE
nu
maxit
tol
```

```
cov
center
wt          wt
n.obs
cor          cor = TRUE
call
iter
```

[covcov.wtcov.mve](#)

```
cov.trob(stackloss)
```

cpus

cpus

name
syct
mmin
mmax
cach
chmin
chmax
perf
estperf

crabs

crabs

crabs

sp species"B"0"
sex
index 1:50
FL
RW
CL
CW
BD

Cushings

Cushings

Cushings

Tetrahydrocortisone

Pregnanetriol

Type abcu

DDT

DDT

deaths

deaths

deaths

ldeaths

denumerate

loglmdenumerateterms

denumerate(x)

x loglm

loglmterms1denumeraten.vnterms

n.vnterms

renumerate

denumerate(~(1+2+3)^3 + a/b)
which gives ~ (.v1 + .v2 + .v3)^3 + a/b

dose.p

```
dose.p(obj, cf = 1:2, p = 0.5)
```

```
obj          "glm"
```

```
cf
```

```
p
```

```
"glm.dose" "p" "SE"
```

```
ldose <- rep(0:5, 2)
```

```
numdead <- c(1, 4, 9, 13, 18, 20, 0, 2, 6, 10, 12, 16)
```

```
sex <- factor(rep(c("M", "F"), c(6, 6)))
```

```
SF <- cbind(numdead, numalive = 20 - numdead)
```

```
budworm.lg0 <- glm(SF ~ sex + ldose - 1, family = binomial)
```

```
dose.p(budworm.lg0, cf = c(1,3), p = 1:3/4)
```

```
dose.p(update(budworm.lg0, family = binomial(link=probit)),  
        cf = c(1,3), p = 1:3/4)
```

drivers

```
drivers
```

dropterm

lmglm

dropterm (object, ...)

Default S3 method:

```
dropterm(object, scope, scale = 0, test = c("none", "Chisq"),  
         k = 2, sorted = FALSE, trace = FALSE, ...)
```

S3 method for class 'lm'

```
dropterm(object, scope, scale = 0, test = c("none", "Chisq", "F"),  
         k = 2, sorted = FALSE, ...)
```

S3 method for class 'glm'

```
dropterm(object, scope, scale = 0, test = c("none", "Chisq", "F"),  
         k = 2, sorted = FALSE, trace = FALSE, ...)
```

object

scope

scale lmaovglmscale

test lmaovglm1m

k k = 2k = log(n)

sorted

trace TRUE

...

lm

"anova"

[addtermstepAIC](#)

```

quine.hi <- aov(log(Days + 2.5) ~ .^4, quine)
quine.nxt <- update(quine.hi, . ~ . - Eth:Sex:Age:Lrn)
dropterm(quine.nxt, test= "F")
quine.stp <- stepAIC(quine.nxt,
  scope = list(upper = ~Eth*Sex*Age*Lrn, lower = ~1),
  trace = FALSE)
dropterm(quine.stp, test = "F")
quine.3 <- update(quine.stp, . ~ . - Eth:Age:Lrn)
dropterm(quine.3, test = "F")
quine.4 <- update(quine.3, . ~ . - Eth:Age)
dropterm(quine.4, test = "F")
quine.5 <- update(quine.4, . ~ . - Age:Lrn)
dropterm(quine.5, test = "F")

house.glm0 <- glm(Freq ~ Infl*Type*Cont + Sat, family=poisson,
  data = housing)
house.glm1 <- update(house.glm0, . ~ . + Sat*(Infl+Type+Cont))
dropterm(house.glm1, test = "Chisq")

```

eagles

eagles

eagles

y

n

P LS

A IA

V LS

```

eagles.glm <- glm(cbind(y, n - y) ~ P*A + V, data = eagles,
  family = binomial)
dropterm(eagles.glm)
prof <- profile(eagles.glm)
plot(prof)
pairs(prof)

```

epil

epil

```
y
trt "placebo""progabide"
base
age
V4 0/1
subject
period
lbase
lage
```

y2123

```
summary(glm(y ~ lbase*trt + lage + V4, family = poisson,
            data = epil), correlation = FALSE)
epil2 <- epil[epil$period == 1, ]
epil2["period"] <- rep(0, 59); epil2["y"] <- epil2["base"]
epil["time"] <- 1; epil2["time"] <- 4
epil2 <- rbind(epil, epil2)
epil2$pred <- unclass(epil2$trt) * (epil2$period > 0)
epil2$subject <- factor(epil2$subject)
epil3 <- aggregate(epil2, list(epil2$subject, epil2$period > 0),
  function(x) if(is.numeric(x)) sum(x) else x[1])
epil3$pred <- factor(epil3$pred,
  labels = c("base", "placebo", "drug"))

contrasts(epil3$pred) <- structure(contr.sdif(3),
  dimnames = list(NULL, c("placebo-base", "drug-placebo")))
## IGNORE_RDIFF_BEGIN
summary(glm(y ~ pred + factor(subject) + offset(log(time)),
```

```

        family = poisson, data = epil3), correlation = FALSE)
## IGNORE_RDIFF_END

summary(glmPQL(y ~ lbase*trt + lage + V4,
               random = ~ 1 | subject,
               family = poisson, data = epil))
summary(glmPQL(y ~ pred, random = ~1 | subject,
               family = poisson, data = epil3))

```

eqscplot

```
eqscplot(x, y, ratio = 1, tol = 0.04, uin, ...)
```

```

x          xy
y
ratio      ratiouin
tol
uin
...        plotpar(xaxs="i", yaxs="i")xlimylim

```

```
ratio1 + tol
```

```
uin
```

```

plot.windowaspratioplot
ratiouin

```

```
plotpar
```

farms

farms

farms

Mois

Manag SFBFHFNM

Use U1U2U3

Manure C0C4

```
farms.mca <- mca(farms, abbrev = TRUE) # Use levels as names
eqscplot(farms.mca$cs, type = "n")
text(farms.mca$rs, cex = 0.7)
text(farms.mca$cs, labels = dimnames(farms.mca$cs)[[1]], cex = 0.7)
```

fgl

fgl

fgl

RI

Na
Mg
Al
Si
K
Ca
Ba
Fe
type WinFWinNFVehConTablHead

fitdistr

fitdistr(x, densfun, start, ...)

x

densfun

"beta" "cauchy" "chi-squared" "exponential" "gamma" "geometric"
"log-normal" "lognormal" "logistic" "negative binomial" "normal"
"Poisson" "t" "weibull"

start

... densfunoptimlowerupperdensfun

start

[optim](#)lowerupperL-BFGS-Bmethod

"t"ms

start"cauchy" "gamma" "logistic" "negative binomial"musize"t" "weibull"

[printcoefvcovlogLik](#)"fitdistr"

"fitdistr"

estimate

sd

vcov

loglik

optimparscale

```
## avoid spurious accuracy
op <- options(digits = 3)
set.seed(123)
x <- rgamma(100, shape = 5, rate = 0.1)
fitdistr(x, "gamma")
## now do this directly with more control.
fitdistr(x, dgamma, list(shape = 1, rate = 0.1), lower = 0.001)

set.seed(123)
x2 <- rt(250, df = 9)
fitdistr(x2, "t", df = 9)
## allow df to vary: not a very good idea!
fitdistr(x2, "t")
## now do fixed-df fit directly with more control.
mydt <- function(x, m, s, df) dt((x-m)/s, df)/s
fitdistr(x2, mydt, list(m = 0, s = 1), df = 9, lower = c(-Inf, 0))

set.seed(123)
x3 <- rweibull(100, shape = 4, scale = 100)
fitdistr(x3, "weibull")

set.seed(123)
x4 <- rnegbin(500, mu = 5, theta = 4)
fitdistr(x4, "Negative Binomial")
options(op)
```

forbes

forbes

bp
pres

fractions

```
fractions(x, cycles = 10, max.denominator = 2000, ...)
```

```
as.fractions(x)
```

```
is.fractions(f)
```

```
x
```

```
cycles
```

```
max.denominator
```

```
max.denominator
```

```
...
```

```
f
```

```
x = floor(x) + 1/(p1 + 1/(p2 + ...))
```

```
p1p2cyclespj > max.denominator
```

```
"fracs"
```

```
"fractions"
```

```
"fractions".Datax"fracs""fractions"
```

rational

```
X <- matrix(runif(25), 5, 5)
```

```
zapsmall(solve(X, X/5)) # print near-zeroes as zero
```

```
fractions(solve(X, X/5))
```

```
fractions(solve(X, X/5)) + 1
```

GAGurine

GAGurine

Age

GAG

galaxies

unfilled

galaxies

```
gal <- galaxies/1000
c(width.SJ(gal, method = "dpi"), width.SJ(gal))
plot(x = c(0, 40), y = c(0, 0.3), type = "n", bty = "l",
      xlab = "velocity of galaxy (1000km/s)", ylab = "density")
rug(gal)
lines(density(gal, width = 3.25, n = 200), lty = 1)
lines(density(gal, width = 2.56, n = 200), lty = 3)
```

`gamma.dispersion`

`gamma.shape``gamma.dispersion(object, ...)``object``... gamma.shape``gamma.shape.glm`

`gamma.shape`

`Gamma``gamma.shape(object, ...)`

```
## S3 method for class 'glm'
gamma.shape(object, it.lim = 10,
             eps.max = .Machine$double.eps^0.25, verbose = FALSE, ...)
```

`object Gammaquasivariance = "mu^2"``it.lim``eps.max``verbose TRUE``...`

alpha

SE

`gamma.dispersion`

```
clotting <- data.frame(
  u = c(5,10,15,20,30,40,60,80,100),
  lot1 = c(118,58,42,35,27,25,21,19,18),
  lot2 = c(69,35,26,21,18,16,13,12,12))
clot1 <- glm(lot1 ~ log(u), data = clotting, family = Gamma)
gamma.shape(clot1)

gm <- glm(Days + 0.1 ~ Age*Eth*Sex*Lrn,
  quasi(link=log, variance="mu^2"), quine,
  start = c(3, rep(0,31)))
gamma.shape(gm, verbose = TRUE)
summary(gm, dispersion = gamma.dispersion(gm)) # better summary
```

gehan

gehan

pair

time

cens

treat

```
library(survival)
gehan.surv <- survfit(Surv(time, cens) ~ treat, data = gehan,
  conf.type = "log-log")
summary(gehan.surv)
survreg(Surv(time, cens) ~ factor(pair) + treat, gehan, dist = "exponential")
summary(survreg(Surv(time, cens) ~ treat, gehan, dist = "exponential"))
summary(survreg(Surv(time, cens) ~ treat, gehan))
gehan.cox <- coxph(Surv(time, cens) ~ treat, gehan)
summary(gehan.cox)
```

genotype

ABIJ

genotype

Litter

Mother

Wt

geyser

geyser

duration
waiting

waiting

faithful

gilgais

gilgais

pH00

pH30

pH80

e00

e30

e80

c00

c30

c80

`ginv`

`X`

`ginv(X, tol = sqrt(.Machine$double.eps))`

`X`

`tol`

`X`

[`solvesvdeigen`](#)

`glm.convert`

`glm.nb()glm()`

`glm.convert(object)`

`object` `"negbin"`[`glm.nb\(\)`](#)

`"glm"`

[`glm.negative.binomialglm`](#)

```
quine.nb1 <- glm.nb(Days ~ Sex/(Age + Eth*Lrn), data = quine)
quine.nbA <- glm.convert(quine.nb1)
quine.nbB <- update(quine.nb1, . ~ . + Sex:Age:Lrn)
anova(quine.nbA, quine.nbB)
```

glm.nb

[glm\(\)](#)theta

```
glm.nb(formula, data, weights, subset, na.action,  
       start = NULL, etastart, mustart,  
       control = glm.control(...), method = "glm.fit",  
       model = TRUE, x = FALSE, y = TRUE, contrasts = NULL, ...,  
       init.theta, link = log)
```

formula data weight subset na.action start etastart mustart control method model
xy contrasts...

[glm\(\)](#)family offset offset()

init.theta

link logsqrtidentity

theta glm() theta theta maxit glm.control

trace > 0 trace > 1 glm trace > 2 theta

negbin glm lm glm theta SE. theta twologlik

[glmnegative.binomialanova.negbinsummary.negbintheta.md](#)

[simulate](#)

```
quine.nb1 <- glm.nb(Days ~ Sex/(Age + Eth*Lrn), data = quine)  
quine.nb2 <- update(quine.nb1, . ~ . + Sex:Age:Lrn)  
quine.nb3 <- update(quine.nb2, Days ~ .^4)  
anova(quine.nb1, quine.nb2, quine.nb3)
```

`glmmPQL`

```
glmmPQL(fixed, random, family, data, correlation, weights,
        control, niter = 10, verbose = TRUE, ...)
```

```
fixed
random
family
data          weights...subset
correlation
weights       glm
control       lme
niter
verbose
...           lme
```

```
glmmPQLlmenlme
lmeoffsetfixedlme
"lme"predict
```

```
c("glmmPQL", "lme")lmeObject
```

`lme`

```
summary(glmmPQL(y ~ trt + I(week > 2), random = ~ 1 | ID,
               family = binomial, data = bacteria))
```

```
## an example of an offset: the coefficient of 'week' changes by one.
summary(glmmPQL(y ~ trt + week, random = ~ 1 | ID,
               family = binomial, data = bacteria))
summary(glmmPQL(y ~ trt + week + offset(week), random = ~ 1 | ID,
               family = binomial, data = bacteria))
```

hills

hills

dist
climb
time

hist.scott

hist.scott(x, prob = TRUE, xlab = deparse(substitute(x)), ...)
hist.FD(x, prob = TRUE, xlab = deparse(substitute(x)), ...)

x
prob
xlab... hist

nclass.*

hist

housing

housing

housing

Sat

Infl

Type

Cont

Freq

```
options(contrasts = c("contr.treatment", "contr.poly"))
```

```
# Surrogate Poisson models
```

```
house.glm0 <- glm(Freq ~ Infl*Type*Cont + Sat, family = poisson,  
                  data = housing)
```

```
## IGNORE_RDIFF_BEGIN
```

```
summary(house.glm0, correlation = FALSE)
```

```
## IGNORE_RDIFF_END
```

```
addterm(house.glm0, ~. + Sat:(Infl+Type+Cont), test = "Chisq")
```

```
house.glm1 <- update(house.glm0, . ~ . + Sat*(Infl+Type+Cont))
```

```
summary(house.glm1, correlation = FALSE)
```

```
1 - pchisq(deviance(house.glm1), house.glm1$df.residual)
```

```
dropterm(house.glm1, test = "Chisq")
```

```
addterm(house.glm1, ~. + Sat:(Infl+Type+Cont)^2, test = "Chisq")
```

```
hnames <- lapply(housing[, -5], levels) # omit Freq
```

```
newData <- expand.grid(hnames)
```

```
newData$Sat <- ordered(newData$Sat)
```

```
house.pm <- predict(house.glm1, newData,  
                    type = "response") # poisson means
```

```
house.pm <- matrix(house.pm, ncol = 3, byrow = TRUE,  
                   dimnames = list(NULL, hnames[[1]]))
```

```

house.pr <- house.pm/drop(house.pm %*% rep(1, 3))
cbind(expand.grid(hnames[-1]), round(house.pr, 2))

# Iterative proportional scaling
loglm(Freq ~ Infl*Type*Cont + Sat*(Infl+Type+Cont), data = housing)

# multinomial model
library(nnet)
(house.mult<- multinom(Sat ~ Infl + Type + Cont, weights = Freq,
  data = housing))
house.mult2 <- multinom(Sat ~ Infl*Type*Cont, weights = Freq,
  data = housing)
anova(house.mult, house.mult2)

house.pm <- predict(house.mult, expand.grid(hnames[-1]), type = "probs")
cbind(expand.grid(hnames[-1]), round(house.pm, 2))

# proportional odds model
house.cpr <- apply(house.pr, 1, cumsum)
logit <- function(x) log(x/(1-x))
house.ld <- logit(house.cpr[, 2, ]) - logit(house.cpr[, 1, ])
(ratio <- sort(drop(house.ld)))
mean(ratio)

(house.plr <- polr(Sat ~ Infl + Type + Cont,
  data = housing, weights = Freq))

house.pr1 <- predict(house.plr, expand.grid(hnames[-1]), type = "probs")
cbind(expand.grid(hnames[-1]), round(house.pr1, 2))

Fr <- matrix(housing$Freq, ncol = 3, byrow = TRUE)
2*sum(Fr*log(house.pr/house.pr1))

house.plr2 <- stepAIC(house.plr, ~.^2)
house.plr2$anova

```

huber

```
huber(y, k = 1.5, tol = 1e-06)
```

```

y
k          k
tol

```

```

mu
s

```

[hubersmad](#)

huber(chem)

hubers

```
hubers(y, k = 1.5, mu, s, initmu = median(y), tol = 1e-06)
```

```
y
k          k
mu
s
initmu     mu
tol
```

```
mu
s
```

[huber](#)

```
hubers(chem)
hubers(chem, mu=3.68)
```

immer

immer

immer

Loc

Var "manchuria""svansota""velvet""trebi""peatland"

Y1

Y2

```
immer.aov <- aov(cbind(Y1,Y2) ~ Loc + Var, data = immer)
summary(immer.aov)
```

```
immer.aov <- aov((Y1+Y2)/2 ~ Var + Loc, data = immer)
summary(immer.aov)
model.tables(immer.aov, type = "means", se = TRUE, cterms = "Var")
```

Insurance

Insurance

Insurance

District

Group

Age

Holders

Claims

```
## main-effects fit as Poisson GLM with offset
glm(Claims ~ District + Group + Age + offset(log(Holders)),
     data = Insurance, family = poisson)

# same via loglm
loglm(Claims ~ District + Group + Age + offset(log(Holders)),
      data = Insurance)
```

isoMDS

```
isoMDS(d, y = cmdscale(d, k), k = 2, maxit = 50, trace = TRUE,
      tol = 1e-3, p = 2)
```

```
Shepard(d, x, p = 2)
```

```
d          dist
y          cmdscale
k          cmdscale
maxit
trace      TRUE
tol
p
x
```

$O(n^2)p = 2$

```
points
stress
```

trace

cmdscal^{esammon}

```
swiss.x <- as.matrix(swiss[, -1])
swiss.dist <- dist(swiss.x)
swiss.mds <- isoMDS(swiss.dist)
plot(swiss.mds$points, type = "n")
text(swiss.mds$points, labels = as.character(1:nrow(swiss.x)))
swiss.sh <- Shepard(swiss.dist, swiss.mds$points)
plot(swiss.sh, pch = ".")
lines(swiss.sh$x, swiss.sh$yf, type = "S")
```

kde2d

```
kde2d(x, y, h, n = 25, lims = c(range(x), range(y)))
```

x	
y	
h	^{bandwidth.nrd}
n	
lims	c(xl, xu, yl, yu)

xy	n
z	n[1]n[2]xy

```

attach(geyser)
plot(duration, waiting, xlim = c(0.5,6), ylim = c(40,100))
f1 <- kde2d(duration, waiting, n = 50, lims = c(0.5, 6, 40, 100))
image(f1, zlim = c(0, 0.05))
f2 <- kde2d(duration, waiting, n = 50, lims = c(0.5, 6, 40, 100),
           h = c(width.SJ(duration), width.SJ(waiting)) )
image(f2, zlim = c(0, 0.05))
persp(f2, phi = 30, theta = 20, d = 5)

plot(duration[-272], duration[-1], xlim = c(0.5, 6),
      ylim = c(1, 6), xlab = "previous duration", ylab = "duration")
f1 <- kde2d(duration[-272], duration[-1],
           h = rep(1.5, 2), n = 50, lims = c(0.5, 6, 0.5, 6))
contour(f1, xlab = "previous duration",
        ylab = "duration", levels = c(0.05, 0.1, 0.2, 0.4) )
f1 <- kde2d(duration[-272], duration[-1],
           h = rep(0.6, 2), n = 50, lims = c(0.5, 6, 0.5, 6))
contour(f1, xlab = "previous duration",
        ylab = "duration", levels = c(0.05, 0.1, 0.2, 0.4) )
f1 <- kde2d(duration[-272], duration[-1],
           h = rep(0.4, 2), n = 50, lims = c(0.5, 6, 0.5, 6))
contour(f1, xlab = "previous duration",
        ylab = "duration", levels = c(0.05, 0.1, 0.2, 0.4) )
detach("geyser")

```

lda

```

lda(x, ...)

## S3 method for class 'formula'
lda(formula, data, ..., subset, na.action)

## Default S3 method:
lda(x, grouping, prior = proportions, tol = 1.0e-4,
    method, CV = FALSE, nu, ...)

## S3 method for class 'data.frame'
lda(x, ...)

## S3 method for class 'matrix'
lda(x, grouping, ..., subset, na.action)

```

```

formula      groups ~ x1 + x2 + ...
data         formula
x

```



```
tol^2
priorpredict.lda
```

```
subset=na.action=
update()
```

[illegible]

ldahist

```
ldahist(data, g, nbins = 25, h, x0 = - h/1000, breaks,  
        xlim = range(breaks), ymax = 0, width,  
        type = c("histogram", "density", "both"),  
        sep = (type != "density"),  
        col = 5, xlab = deparse(substitute(data)), bty = "n", ...)
```

data	NA
g	data
nbins	
h	nbins
x0	$x0 + h * (... , -1, 0, 1, ...)$
breaks	hnbins
xlim	
ymax	
width	
type	
sep	
col	
xlab	data
bty	
...	polygon

[plot.lda](#)

leuk

leuk

wbc

ag "present""absent"

time

```
library(survival)
plot(survfit(Surv(time) ~ ag, data = leuk), lty = 2:3, col = 2:3)

# now Cox models
leuk.cox <- coxph(Surv(time) ~ ag + log(wbc), leuk)
summary(leuk.cox)
```

lm.gls

```
lm.gls(formula, data, W, subset, na.action, inverse = FALSE,
        method = "qr", model = FALSE, x = FALSE, y = FALSE,
        contrasts = NULL, ...)
```

formula	response ~ predictorsformula
data	formula
W	
subset	
na.action	
inverse	W
method	lm.fit
model	
x	
y	
contrasts	
...	lm.fit

lm.fit

"lm.gls""lm""weights""lm"

glslmlm.ridge

lm.ridge

```
lm.ridge(formula, data, subset, na.action, lambda = 0, model = FALSE,
          x = FALSE, y = FALSE, contrasts = NULL, ...)
select(obj)
```

formula	response ~ predictorsformulaoffset
data	formula
subset	
na.action	
lambda	
model	
x	
y	
contrasts	contrasts.argmodel.matrix.default
...	lm.fit
obj	"lm.ridge"

```
coef          lambdacoef
scales
Inter
lambda
ym            y
xm            x
GCV
kHKB
kLW
```

lm

```
longley # not the same as the S-PLUS dataset
names(longley)[1] <- "y"
lm.ridge(y ~ ., longley)
plot(lm.ridge(y ~ ., longley,
              lambda = seq(0,0.1,0.001)))
select(lm.ridge(y ~ ., longley,
                lambda = seq(0,0.1,0.001)))
```

loglm

loglinglm

```
loglm(formula, data, subset, na.action, ...)
```

```
formula
          datadimnames.
data
          xtabs
subset
na.action
...          loglm1
```

```
dataloglinstartloglin
```

```
loglina:ba*ba/b
```

```
"loglm"printsummarydeviancefittedcoefresidanovaupdateanova
```

```
loglinloglm
```

```
loglm1loglin
```

```
# The data frames Cars93, minn38 and quine are available
# in the MASS package.

# Case 1: frequencies specified as an array.
sapply(minn38, function(x) length(levels(x)))
## hs phs fol sex f
## 3 4 7 2 0
##minn38a <- array(0, c(3,4,7,2), lapply(minn38[, -5], levels))
##minn38a[data.matrix(minn38[, -5])] <- minn38$f

## or more simply
minn38a <- xtabs(f ~ ., minn38)

fm <- loglm(~ 1 + 2 + 3 + 4, minn38a) # numerals as names.
deviance(fm)
## [1] 3711.9
fm1 <- update(fm, .~.^2)
fm2 <- update(fm, .~.^3, print = TRUE)
## 5 iterations: deviation 0.075
anova(fm, fm1, fm2)

# Case 1. An array generated with xtabs.

loglm(~ Type + Origin, xtabs(~ Type + Origin, Cars93))

# Case 2. Frequencies given as a vector in a data frame
names(quine)
## [1] "Eth" "Sex" "Age" "Lrn" "Days"
fm <- loglm(Days ~ .^2, quine)
gm <- glm(Days ~ .^2, poisson, quine) # check glm.
c(deviance(fm), deviance(gm)) # deviances agree
## [1] 1368.7 1368.7
c(fm$df, gm$df) # resid df do not!
c(fm$df, gm$df.residual) # resid df do not!
## [1] 127 128
# The loglm residual degrees of freedom is wrong because of
# a non-detectable redundancy in the model matrix.
```

logtrans

$\log(y + \alpha) \sim x_1 + x_2 + \dots$

`logtrans(object, ...)`

Default S3 method:

```
logtrans(object, ..., alpha = seq(0.5, 6, by = 0.25) - min(y),
        plotit = TRUE, interp =, xlab = "alpha",
        ylab = "log Likelihood")
```

S3 method for class 'formula'

```
logtrans(object, data, ...)
```

S3 method for class 'lm'

```
logtrans(object, ...)
```

`object` `y ~ x1 + x2 + ...`

`...` `objectlm`

`alpha`

`plotit`

`interp` `TRUE`

`xlab` `plot`

`ylab` `plot`

`data` `datalm`

`xy`

[boxcox](#)

```
logtrans(Days ~ Age*Sex*Eth*Lrn, data = quine,
        alpha = seq(0.75, 6.5, length.out = 20))
```

lqs

lmsregltsreg

lqs(x, ...)

S3 method for class 'formula'

```
lqs(formula, data, ...,
    method = c("lts", "lqs", "lms", "S", "model.frame"),
    subset, na.action, model = TRUE,
    x.ret = FALSE, y.ret = FALSE, contrasts = NULL)
```

Default S3 method:

```
lqs(x, y, intercept = TRUE, method = c("lts", "lqs", "lms", "S"),
    quantile, control = lqs.control(...), k0 = 1.548, seed, ...)
```

lmsreg(...)

ltsreg(...)

formula y ~ x1 + x2 + ...

data formula

subset

na.action NA
na.omit
na.exclude

modelx.rety.ret

TRUE

contrasts contrasts.arg
model.matrix.default

x

y x

intercept

method model.frameDetailslmsregltsreg"lms""lts"

quantile Detailsmethod = "lms"

control Details

k0 $\chi(\psi)$ method = "S"

seed .Random.seed.Random.seed

... lqs.defaultlqs.controlcontrolDetails

np

```
"lqs""lms"quantile"lts"quantile"lqs""lms"quantilefloor((n+p+1)/2)floor((n+1)/2)
"lts"floor(n/2) + floor((p+1)/2)
```

"S"ss

control

psamp p

nsamp "best""exact""sample""sample"min(5*p, 3000)"best""exact"

adjust TRUE


```

"lqs"
crit          method == "S"
sing
coefficients
bestone
fitted.values
residuals
scale          method == "S"

lmslqsn-1/3(floor((n-p)/2) + 1)/nfloor((n+p)/2) <= quantile <= floor((n+p+1)/2)
np
psamp

```

[predict.lqs](#)

```

## IGNORE_RDIFF_BEGIN
set.seed(123) # make reproducible
lqs(stack.loss ~ ., data = stackloss)
lqs(stack.loss ~ ., data = stackloss, method = "S", nsamp = "exact")
## IGNORE_RDIFF_END

```

mammals

mammals

body
brain
name

mca

```
mca(df, nf = 2, abbrev = FALSE)
```

```
df
nf
abbrev      abbrev = TRUE
```

```
"mca"
rs      nf
cs
fs      predict.mca
p
d      nf
call
```

[predict.mcaplot.mcacoresp](#)

```
farms.mca <- mca(farms, abbrev=TRUE)
farms.mca
plot(farms.mca)
```

mcycle

```
mcycle
```

```
times
accel
```

Melanoma

Melanoma

Melanoma

time
status 123
sex 10
age
year
thickness
ulcer 10

menarche

menarche

Age
Total
Menarche

```
mprob <- glm(cbind(Menarche, Total - Menarche) ~ Age,  
             binomial(link = probit), data = menarche)
```

micelson

micelson

Expt
Run
Speed

minn38

minn38

minn38

hs "L""M""U"
phs "C""N""E""O"
fo1 "F1""F2""F7"
sex "F""M"
f

motors

motors

motors

temp

time

cens

```
library(survival)
plot(survfit(Surv(time, cens) ~ factor(temp), motors), conf.int = FALSE)
# fit Weibull model
motor.wei <- survreg(Surv(time, cens) ~ temp, motors)
## IGNORE_RDIFF_BEGIN
summary(motor.wei)
## IGNORE_RDIFF_END
# and predict at 130C
unlist(predict(motor.wei, data.frame(temp=130), se.fit = TRUE))

motor.cox <- coxph(Surv(time, cens) ~ temp, motors)
summary(motor.cox)
# predict at temperature 200
plot(survfit(motor.cox, newdata = data.frame(temp=200),
  conf.type = "log-log"))
summary( survfit(motor.cox, newdata = data.frame(temp=130)) )
```

muscle

muscle

Strip

Conc

Length

```
## IGNORE_RDIFF_BEGIN
A <- model.matrix(~ Strip - 1, data=muscle)
rats.nls1 <- nls(log(Length) ~ cbind(A, rho^Conc),
  data = muscle, start = c(rho=0.1), algorithm="plinear")
(B <- coef(rats.nls1))

st <- list(alpha = B[2:22], beta = B[23], rho = B[1])
(rats.nls2 <- nls(log(Length) ~ alpha[Strip] + beta*rho^Conc,
  data = muscle, start = st))
## IGNORE_RDIFF_END

Muscle <- with(muscle, {
  Muscle <- expand.grid(Conc = sort(unique(Conc)), Strip = levels(Strip))
  Muscle$Yhat <- predict(rats.nls2, Muscle)
  Muscle <- cbind(Muscle, logLength = rep(as.numeric(NA), 126))
  ind <- match(paste(Strip, Conc),
    paste(Muscle$Strip, Muscle$Conc))
  Muscle$logLength[ind] <- log(Length)
  Muscle})

lattice::xyplot(Yhat ~ Conc | Strip, Muscle, as.table = TRUE,
  ylim = range(c(Muscle$Yhat, Muscle$logLength), na.rm = TRUE),
  subscripts = TRUE, xlab = "Calcium Chloride concentration (mM)",
  ylab = "log(Length in mm)", panel =
  function(x, y, subscripts, ...) {
    panel.xyplot(x, Muscle$logLength[subscripts], ...)
    llines(spline(x, y))
  })
```

`mvrnorm`

```
mvrnorm(n = 1, mu, Sigma, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
```

`n`

`mu`

`Sigma`

`tol` `Sigma`

`empirical`

`EISPACK` `FALSE`

`eigen`

```
n = 1*length(mu)
```

```
.Random.seed
```

`rnorm`

```
Sigma <- matrix(c(10,3,3,2),2,2)
```

```
Sigma
```

```
var(mvrnorm(n = 1000, rep(0, 2), Sigma))
```

```
var(mvrnorm(n = 1000, rep(0, 2), Sigma, empirical = TRUE))
```

```
negative.binomial
```

```
thetaglm()
```

```
negative.binomial(theta = stop("'theta' must be specified"), link = "log")
```

```
theta      theta
link       logsqrtidentity"link-glm"
```

```
"family"glm()
```

```
glm.nbanova.negbinsummary.negbin
```

```
# Fitting a Negative Binomial model to the quine data
#   with theta = 2 assumed known.
#
glm(Days ~ .^4, family = negative.binomial(2), data = quine)
```

```
newcomb
```

```
28 26 33 24 34 -44 27 16 40 -2 29 22 24 21 25 30 23 29 31 19
24 20 36 32 36 28 25 21 28 29 37 25 28 26 30 32 36 26 30 22
36 23 27 27 28 27 31 27 26 33 26 32 32 24 39 28 24 25 32 25
29 27 28 29 16 23
```

```
newcomb
```

nlschools

nlschools

lang

IQ

class

GS COMB

SES

COMB 0/11

```
n11 <- within(nlschools, {  
  IQave <- tapply(IQ, class, mean)[as.character(class)]  
  IQ <- IQ - IQave  
})  
cen <- c("IQ", "IQave", "SES")  
n11[cen] <- scale(n11[cen], center = TRUE, scale = FALSE)  
  
n1.lme <- nlme::lme(lang ~ IQ*COMB + IQave + SES,  
                   random = ~ IQ | class, data = n11)  
## IGNORE_RDIFF_BEGIN  
summary(n1.lme)  
## IGNORE_RDIFF_END
```

npk

npk

npk

block

N

P

K

yield

```
options(contrasts = c("contr.sum", "contr.poly"))
npk.aov <- aov(yield ~ block + N*P*K, npk)
## IGNORE_RDIFF_BEGIN
npk.aov
summary(npk.aov)
alias(npk.aov)
coef(npk.aov)
options(contrasts = c("contr.treatment", "contr.poly"))
npk.aov1 <- aov(yield ~ block + N + K, data = npk)
summary.lm(npk.aov1)
se.contrast(npk.aov1, list(N=="0", N=="1"), data = npk)
model.tables(npk.aov1, type = "means", se = TRUE)
## IGNORE_RDIFF_END
```

npr1

npr1

x
y
perm
por

Null

MNcrossprod(N, M) = t(N) %*% MN

Null(M)

M

$\{x : Mx = 0\}$ Null(t(M))

NM

qrqr.Q

```
# The function is currently defined as
function(M)
{
  tmp <- qr(M)
  set <- if(tmp$rank == 0L) seq_len(ncol(M)) else -seq_len(tmp$rank)
  qr.Q(tmp, complete = TRUE)[, set, drop = FALSE]
}
```

oats

oats

B

V

N

Y

```
oats$Nf <- ordered(oats$N, levels = sort(levels(oats$N)))
oats.aov <- aov(Y ~ Nf*V + Error(B/V), data = oats, qr = TRUE)
## IGNORE_RDIFF_BEGIN
summary(oats.aov)
summary(oats.aov, split = list(Nf=list(L=1, Dev=2:3)))
## IGNORE_RDIFF_END
par(mfrow = c(1,2), pty = "s")
plot(fitted(oats.aov[[4]]), studres(oats.aov[[4]]))
abline(h = 0, lty = 2)
oats.pr <- proj(oats.aov)
qqnorm(oats.pr[[4]][,"Residuals"], ylab = "Stratum 4 residuals")
qqline(oats.pr[[4]][,"Residuals"])

par(mfrow = c(1,1), pty = "m")
oats.aov2 <- aov(Y ~ N + V + Error(B/V), data = oats, qr = TRUE)
model.tables(oats.aov2, type = "means", se = TRUE)
```

OME

OME

OME

ID

OME "low""high""N/A"

Age

Loud

Noise "coherent""incoherent"

Correct Trials

Trials

```
# Fit logistic curve from p = 0.5 to p = 1.0
fp1 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/scal)),
             c("L75", "scal"),
             function(x,L75,scal)NULL)
nls(Correct/Trials ~ fp1(Loud, L75, scal), data = OME,
    start = c(L75=45, scal=3))
nls(Correct/Trials ~ fp1(Loud, L75, scal),
    data = OME[OME$Noise == "coherent",],
    start=c(L75=45, scal=3))
nls(Correct/Trials ~ fp1(Loud, L75, scal),
    data = OME[OME$Noise == "incoherent",],
    start = c(L75=45, scal=3))

# individual fits for each experiment

aa <- factor(OME$Age)
ab <- 10*OME$ID + unclass(aa)
ac <- unclass(factor(ab))
OME$UID <- as.vector(ac)
OME$UIDn <- OME$UID + 0.1*(OME$Noise == "incoherent")
rm(aa, ab, ac)
```

```

OMEi <- OME

library(nlme)
fp2 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/2)),
             "L75", function(x,L75) NULL)
dec <- getOption("OutDec")
options(show.error.messages = FALSE, OutDec=".")
OMEi.nls <- nlsList(Correct/Trials ~ fp2(Loud, L75) | UIDn,
                   data = OMEi, start = list(L75=45), control = list(maxiter=100))
options(show.error.messages = TRUE, OutDec=dec)
tmp <- sapply(OMEi.nls, function(X)
              {if(is.null(X)) NA else as.vector(coef(X))})
OMEif <- data.frame(UID = round(as.numeric((names(tmp))))),
                  Noise = rep(c("coherent", "incoherent"), 110),
                  L75 = as.vector(tmp), stringsAsFactors = TRUE)
OMEif$Age <- OME$Age[match(OMEif$UID, OME$UID)]
OMEif$OME <- OME$OME[match(OMEif$UID, OME$UID)]
OMEif <- OMEif[OMEif$L75 > 30,]
summary(lm(L75 ~ Noise/Age, data = OMEif, na.action = na.omit))
summary(lm(L75 ~ Noise/(Age + OME), data = OMEif,
            subset = (Age >= 30 & Age <= 60),
            na.action = na.omit), correlation = FALSE)

# Or fit by weighted least squares
fpl75 <- deriv(~ sqrt(n)*(r/n - 0.5 - 0.5/(1 + exp(-(x-L75)/scal))),
              c("L75", "scal"),
              function(r,n,x,L75,scal) NULL)
nls(0 ~ fpl75(Correct, Trials, Loud, L75, scal),
    data = OME[OME$Noise == "coherent",],
    start = c(L75=45, scal=3))
nls(0 ~ fpl75(Correct, Trials, Loud, L75, scal),
    data = OME[OME$Noise == "incoherent",],
    start = c(L75=45, scal=3))

# Test to see if the curves shift with age
fpl75age <- deriv(~sqrt(n)*(r/n - 0.5 - 0.5/(1 +
exp(-(x-L75-slope*age)/scal))),
                 c("L75", "slope", "scal"),
                 function(r,n,x,age,L75,slope,scal) NULL)
OME.nls1 <-
nls(0 ~ fpl75age(Correct, Trials, Loud, Age, L75, slope, scal),
    data = OME[OME$Noise == "coherent",],
    start = c(L75=45, slope=0, scal=2))
sqrt(diag(vcov(OME.nls1)))

OME.nls2 <-
nls(0 ~ fpl75age(Correct, Trials, Loud, Age, L75, slope, scal),
    data = OME[OME$Noise == "incoherent",],
    start = c(L75=45, slope=0, scal=2))
sqrt(diag(vcov(OME.nls2)))

# Now allow random effects by using NLME
OMEf <- OME[rep(1:nrow(OME), OME$Trials),]
OMEf$Resp <- with(OME, rep(rep(c(1,0), length(Trials)),
                           t(cbind(Correct, Trials-Correct))))
OMEf <- OMEf[, -match(c("Correct", "Trials"), names(OMEf))]

```

```
## Not run: ## these fail in R on most platforms
fp2 <- deriv(~ 0.5 + 0.5/(1 + exp(-(x-L75)/exp(lsc))),
             c("L75", "lsc"),
             function(x, L75, lsc) NULL)
try(summary(nlme(Resp ~ fp2(Loud, L75, lsc),
                 fixed = list(L75 ~ Age, lsc ~ 1),
                 random = L75 + lsc ~ 1 | UID,
                 data = OMEf[OMEf$Noise == "coherent",], method = "ML",
                 start = list(fixed=c(L75=c(48.7, -0.03), lsc=0.24)), verbose = TRUE)))

try(summary(nlme(Resp ~ fp2(Loud, L75, lsc),
                 fixed = list(L75 ~ Age, lsc ~ 1),
                 random = L75 + lsc ~ 1 | UID,
                 data = OMEf[OMEf$Noise == "incoherent",], method = "ML",
                 start = list(fixed=c(L75=c(41.5, -0.1), lsc=0)), verbose = TRUE)))

## End(Not run)
```

painters

painters

Composition

Drawing

Colour

Expression

School "A""B""C""D""E""F""G""H"

pairs.lda

```
## S3 method for class 'lda'
pairs(x, labels = colnames(x), panel = panel.lda,
      dimen, abbrev = FALSE, ..., cex=0.7, type = c("std", "trellis"))
```

```
x          "lda"
labels
panel
dimen      x
abbrev     abbrev > 0minlengthabbreviate
...        pairs.default
cex        cex
type       pairs.default"trellis"splom
```

```
pairs()"lda"pairs(x)xpairs.lda(x)
```

[pairs](#)

parcoord

```
parcoord(x, col = 1, lty = 1, var.label = FALSE, ...)
```

```
x
col
lty
var.label  TRUE
...        matplot
```



```
parcoord(state.x77[, c(7, 4, 6, 2, 5, 3)])

ir <- rbind(iris3[,1], iris3[,2], iris3[,3])
parcoord(log(ir)[, c(3, 4, 2, 1)], col = 1 + (0:149)%/%50)
```

petrol

petrol

No
SG
VP
V10
EP
Y

```
library(nlme)
Petrol <- petrol
Petrol[, 2:5] <- scale(as.matrix(Petrol[, 2:5]), scale = FALSE)
pet3.lme <- lme(Y ~ SG + VP + V10 + EP,
               random = ~ 1 | No, data = Petrol)
pet3.lme <- update(pet3.lme, method = "ML")
pet4.lme <- update(pet3.lme, fixed. = Y ~ V10 + EP)
anova(pet4.lme, pet3.lme)
```

Pima.tr

Pima.tr
Pima.tr2
Pima.te

npreg
glu
bp
skin
bmi ²
ped
age
type YesNo

Pima.trPima.tePima.tr2Pima.tr

plot.lda

```
## S3 method for class 'lda'  
plot(x, panel = panel.lda, ..., cex = 0.7, dimen,  
      abbrev = FALSE, xlab = "LD1", ylab = "LD2")
```

x	"lda"
panel	
...	pairsldahisteqsplot
cex	cex
dimen	x
abbrev	abbrev > 0minlengthabbreviate
xlab	
ylab	

```
plot()"lda"plot(x)xplot.lda(x)
dimendimen > 2pairsdimen = 2dimen = 1type"histogram""density""both"
```

[pairs.lda](#)[ldahist](#)[ldapredict.lda](#)

`plot.mca`

```
## S3 method for class 'mca'
plot(x, rows = TRUE, col, cex = par("cex"), ...)
```

x	"mca"
rows	
col	cex
...	plot

[mcapredict.mca](#)

```
plot(mca(farms, abbrev = TRUE))
```

polr

```
polr(formula, data, weights, start, ..., subset, na.action,
      contrasts = NULL, Hess = FALSE, model = TRUE,
      method = c("logistic", "probit", "loglog", "cloglog", "cauchit"))
```

formula	response ~ predictors	formula
data	formula	
weights		
start	c(coefficients, zeta)	
...	optimcontrol	
subset		
na.action		
contrasts		
Hess	summaryvcov	
model		
method		

$Y_i Y_i$

$$\zeta_0 = -\infty < \zeta_1 < \cdots < \zeta_K = \infty$$

$$P(Y \leq k|x) = \zeta_k - \eta$$

η [eta](#)beta

k [k](#)

$$F^{-1}(p) = -\log(-\log(p))F^{-1}(p) = \log(-\log(1-p))$$

[predictsummaryvcovanovamodel.frameextractAICstepAICstepprofileconfint](#)

"polr"

coefficients

zeta

deviance

fitted.values

lev

terms terms

```
df.residual
edf
nobs          nobsstepAIC
call
method
convergence    optim
niter          optim
lp
Hessian        Hess
model          model
```

```
vcov
```

```
method = "cloglog"
```

```
optimlmmultinom
```

```
options(contrasts = c("contr.treatment", "contr.poly"))
house.plr <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
house.plr
summary(house.plr, digits = 3)
## slightly worse fit from
summary(update(house.plr, method = "probit", Hess = TRUE), digits = 3)
## although it is not really appropriate, can fit
summary(update(house.plr, method = "loglog", Hess = TRUE), digits = 3)
summary(update(house.plr, method = "cloglog", Hess = TRUE), digits = 3)

predict(house.plr, housing, type = "p")
addterm(house.plr, ~.^2, test = "Chisq")
house.plr2 <- stepAIC(house.plr, ~.^2)
house.plr2$anova
anova(house.plr, house.plr2)

house.plr <- update(house.plr, Hess=TRUE)
pr <- profile(house.plr)
confint(pr)
plot(pr)
pairs(pr)
```

`predict.glmmPQL`

```
## S3 method for class 'glmmPQL'
predict(object, newdata = NULL, type = c("link", "response"),
        level, na.action = na.pass, ...)
```

```
object          "glmmPQL"
newdata
type            "response" type = "response"
level
na.action       newdataNA
...
```

```
level
```

[glmmPQLpredict.lme](#)

```
fit <- glmmPQL(y ~ trt + I(week > 2), random = ~1 | ID,
               family = binomial, data = bacteria)
predict(fit, bacteria, level = 0, type="response")
predict(fit, bacteria, level = 1, type="response")
```

`predict.lda`

```
lda
```

```
## S3 method for class 'lda'
predict(object, newdata, prior = object$prior, dimen,
        method = c("plug-in", "predictive", "debiased"), ...)
```

```
object          "lda"
newdata          objectlda
prior            lda
dimen            min(p, ng-1)dimenmethod="predictive"x
method          "plug-in""debiased""predictive"
...
```

```
predict()"lda"predict(x)xpredict.lda(x)
newdataNAnewdatana.action
prior
```

```
class
posterior
x          dimen
```

```
ldaqpdapredict.qda
```

```
tr <- sample(1:50, 25)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
z <- lda(train, cl)
predict(z, test)$class
```

```
predict.lqs
```

```
lqs
```

```
## S3 method for class 'lqs'
predict(object, newdata, na.action = na.pass, ...)
```

```
object          "lqs"
newdata          objectnewdata1qs
na.action        newdataNA
...
```

```
predict()lqspredict(x)xpredict.lqs(x)
newdataNAnewdatana.action
```

[lqs](#)

```
set.seed(123)
fm <- lqs(stack.loss ~ ., data = stackloss, method = "S", nsamp = "exact")
predict(fm, stackloss)
```

`predict.mca`

```
## S3 method for class 'mca'
predict(object, newdata, type = c("row", "factor"), ...)
```

object	"mca"mca
newdata	object
type	
...	predict

```
type = "row"
type = "factor"
```

[mcaplot.mca](#)

predict.qda

qda

```
## S3 method for class 'qda'
predict(object, newdata, prior = object$prior,
        method = c("plug-in", "predictive", "debiased", "looCV"), ...)
```

```
object      "qda"
newdata     objectqda
prior       qda
method      "plug-in""debiased""predictive""looCV"
...
```

```
predict()"qda"predict(x)xpredict.qda(x)
newdataNAnewdatana.action
```

```
class
posterior
```

[qdaldapredict.lda](#)

```
tr <- sample(1:50, 25)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
zq <- qda(train, cl)
predict(zq, test)$class
```

profile.glm

"glm"

qda

```
qda(x, ...)

## S3 method for class 'formula'
qda(formula, data, ..., subset, na.action)

## Default S3 method:
qda(x, grouping, prior = proportions,
    method, CV = FALSE, nu, ...)

## S3 method for class 'data.frame'
qda(x, ...)

## S3 method for class 'matrix'
qda(x, grouping, ..., subset, na.action)


formula      groups ~ x1 + x2 + ...
data          formula
x
grouping
prior
subset
na.action     NA
method        "moment""mle""mve"cov.mve"t"
CV
nu            method = "t"
...


"qda"

prior
means
scaling       iscaling[,i]
ldet
lev
```

terms
call

CV=TRUE

class
posterior

[predict.qdalda](#)

```
tr <- sample(1:50, 25)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
z <- qda(train, cl)
predict(z,test)$class
```

quine

quine

quine

Eth "A""N"
Sex "F""M"
Age "F0""F1","F2""F3"
Lrn "AL""SL"
Days

Rabbit

$5 - HT_3$

Rabbit

BPchange

Dose

Run "C1""C5""M1""M5"

Treatment $5 - HT_3$

Animal "R1""R5"

rational

`rational(x, cycles = 10, max.denominator = 2000, ...)`

`x`

`cycles`

`max.denominator`

`max.denominator`

`...`

`x = floor(x) + 1/(p1 + 1/(p2 + ...))`

`p1p2cyclespj > max.denominator`

x

[fractions](#)

```
X <- matrix(runif(25), 5, 5)
zapsmall(solve(X, X/5)) # print near-zeroes as zero
rational(solve(X, X/5))
```

renumerate

[denumerate](#)[loglm](#)[terms](#)[renumerate](#)

renumerate(x)

x [denumerate](#)

[denumerate](#)[terms](#)

[.vnnn](#)[loglm](#)

[denumerate](#)

```
denumerate(~(1+2+3)^3 + a/b)
## ~ (.v1 + .v2 + .v3)^3 + a/b
renumerate(.Last.value)
## ~ (1 + 2 + 3)^3 + a/b
```

rlm

```
rlm(x, ...)

## S3 method for class 'formula'
rlm(formula, data, weights, ..., subset, na.action,
     method = c("M", "MM", "model.frame"),
     wt.method = c("inv.var", "case"),
     model = TRUE, x.ret = TRUE, y.ret = FALSE, contrasts = NULL)

## Default S3 method:
rlm(x, y, weights, ..., w = rep(1, nrow(x)),
     init = "ls", psi = psi.huber,
     scale.est = c("MAD", "Huber", "proposal 2"), k2 = 1.345,
     method = c("M", "MM"), wt.method = c("inv.var", "case"),
     maxit = 20, acc = 1e-4, test.vec = "resid", lqs.control = NULL)

psi.huber(u, k = 1.345, deriv = 0)
psi.hampel(u, a = 2, b = 4, c = 8, deriv = 0)
psi.bisquare(u, c = 4.685, deriv = 0)
```

formula	y ~ x1 + x2 + ...
data	formula
weights	
subset	
na.action	NA na.omit options (na.action=)
x	
y	x
method	formula
wt.method	
model	
x.ret	
y.ret	
contrasts	lm
w	
init	coef"ls"w*weights"lts"
psi	g(x, ..., deriv)deriv=0deriv=1...
scale.est	"Huber""proposal 2"
k2	

```

maxit
acc
test.vec
...          rlm.defaultpsi
lqs.control  lqs
u
kabc
deriv       01

```

```

psi.huberpsi.hampelpsi.bisquare
method = "MM" k0 = 1.548n >> pc > k0 cmethod = "MM"

```

```

"rlm""lm"df.residualNA"lm"
lm
s
w
psi
conv
converged
wresid      "inv.var"

```

7.3-52formula

[lmlqs](#)

```

summary(rlm(stack.loss ~ ., stackloss))
rlm(stack.loss ~ ., stackloss, psi = psi.hampel, init = "lts")
rlm(stack.loss ~ ., stackloss, psi = psi.bisquare)

```

rms.curv

$c^{\theta} c^{\iota}$

rms.curv(obj)

obj "nls"

deriv3

print.rms.curvpcic

pcic

rms.curvpcicctci $c^{\theta} c^{\iota} C$

deriv3

```
# The treated sample from the Puromycin data
mmcurve <- deriv3(~ Vm * conc/(K + conc), c("Vm", "K"),
                  function(Vm, K, conc) NULL)
Treated <- Puromycin[Puromycin$state == "treated", ]
(Purfit1 <- nls(rate ~ mmcurve(Vm, K, conc), data = Treated,
                start = list(Vm=200, K=0.1)))
rms.curv(Purfit1)
##Parameter effects: c^theta x sqrt(F) = 0.2121
##      Intrinsic: c^iota x sqrt(F) = 0.092
```

rnegbin

$\mu\mu + \mu^2/\theta$

rnegbin(n, mu = n, theta = stop("'theta' must be specified"))

n length(n)nmu

mu

theta theta

rnbinom

.Random.seed

```
# Negative Binomials with means fitted(fm) and theta = 4.5
fm <- glm.nb(Days ~ ., data = quine)
dummy <- rnegbin(fitted(fm), theta = 4.5)
```

road

road

state

deaths

drivers

popden

rural

temp

fuel

rotifer

pmkc

rotifer

density

pm.y

pm.total

kc.y

kc.tot

Rubber

Rubber

loss

hard

tens

sammon

```
sammon(d, y = cmdscale(d, k), k = 2, niter = 100, trace = TRUE,  
       magic = 0.2, tol = 1e-4)
```

d	dist
y	cmdscaled
k	
niter	
trace	TRUE
magic	
tol	

$O(n^2)$ magic

points
stress

[cmdscaleisoMDS](#)

```
swiss.x <- as.matrix(swiss[, -1])  
swiss.sam <- sammon(dist(swiss.x))  
plot(swiss.sam$points, type = "n")  
text(swiss.sam$points, labels = as.character(1:nrow(swiss.x)))
```

ships

ships

type "A""E"

year "60""65""70""75"

period

service

incidents

shoes

shoes

shrimp

shrimp

shuttle

shuttle

shuttle

stability stabxstab
error MMSSLXXL
sign ppnn
wind headtail
magn LightMediumStrongOut of Range
vis yesno
use autonoauto

Sitka

Sitka

Sitka

size
Time
tree
treat "ozone""control"

[Sitka89](#)

Sitka89

Sitka89

Sitka89

size
Time
tree
treat "ozone""control"

Sitka

Skye

Skye

Skye

A
F
M

```

# ternary() is from the on-line answers.
ternary <- function(X, pch = par("pch"), lcex = 1,
                    add = FALSE, ord = 1:3, ...)
{
  X <- as.matrix(X)
  if(any(X < 0)) stop("X must be non-negative")
  s <- drop(X %*% rep(1, ncol(X)))
  if(any(s<=0)) stop("each row of X must have a positive sum")
  if(max(abs(s-1)) > 1e-6) {
    warning("row(s) of X will be rescaled")
    X <- X / s
  }
  X <- X[, ord]
  s3 <- sqrt(1/3)
  if(!add)
  {
    oldpty <- par("pty")
    on.exit(par(pty=oldpty))
    par(pty="s")
    plot(c(-s3, s3), c(0.5-s3, 0.5+s3), type="n", axes=FALSE,
          xlab="", ylab="")
    polygon(c(0, -s3, s3), c(1, 0, 0), density=0)
    lab <- NULL
    if(!is.null(dn <- dimnames(X))) lab <- dn[[2]]
    if(length(lab) < 3) lab <- as.character(1:3)
    eps <- 0.05 * lcex
    text(c(0, s3+eps*0.7, -s3-eps*0.7),
          c(1+eps, -0.1*eps, -0.1*eps), lab, cex=lcex)
  }
  points((X[,2] - X[,3])*s3, X[,1], ...)
}

ternary(Skye/100, ord=c(1,3,2))

```

snails

snails

Species 12
 Exposure
 Rel.Hum
 Temp
 Deaths
 N

SP500

SP500

stdres

stdres(object)

object

[residualsstdres](#)

steam

steam

Temp
Press

stepAIC

```
stepAIC(object, scope, scale = 0,  
        direction = c("both", "backward", "forward"),  
        trace = 1, keep = NULL, steps = 1000, use.start = FALSE,  
        k = 2, ...)
```

object	
scope	upperlower
scale	lmaovextractAIC
direction	"both""backward""forward""both"scopedirection"backward"
trace	stepAIC
keep	AICkeep
steps	
use.start	glm
k	$k = 2k = \log(n)$
...	extractAIC

```
scopelowerupper$scopeupperlower$scopeupper
scopeobjectupdate.formula
glm$scaleglmextractAICgaussianbinomialpoissonsscalescale
lmaovglm
```

```
"anova""keep"keep="Resid. Dev"lmaovsurvreg
```

```
na.actionna.fail
```

```
addtermdroptermstep
```

```
quine.hi <- aov(log(Days + 2.5) ~ .^4, quine)
quine.nxt <- update(quine.hi, . ~ . - Eth:Sex:Age:Lrn)
quine.stp <- stepAIC(quine.nxt,
  scope = list(upper = ~Eth*Sex*Age*Lrn, lower = ~1),
  trace = FALSE)
quine.stp$anova

cpus1 <- cpus
for(v in names(cpus)[2:7])
  cpus1[[v]] <- cut(cpus[[v]], unique(quantile(cpus[[v]])),
    include.lowest = TRUE)
cpus0 <- cpus1[, 2:8] # excludes names, authors' predictions
cpus.samp <- sample(1:209, 100)
cpus.lm <- lm(log10(perf) ~ ., data = cpus1[cpus.samp,2:8])
cpus.lm2 <- stepAIC(cpus.lm, trace = FALSE)
cpus.lm2$anova

example(birthwt)
birthwt.glm <- glm(low ~ ., family = binomial, data = bwt)
birthwt.step <- stepAIC(birthwt.glm, trace = FALSE)
birthwt.step$anova
birthwt.step2 <- stepAIC(birthwt.glm, ~ .^2 + I(scale(age)^2)
  + I(scale(lwt)^2), trace = FALSE)
birthwt.step2$anova

quine.nb <- glm.nb(Days ~ .^4, data = quine)
quine.nb2 <- stepAIC(quine.nb)
quine.nb2$anova
```

stormer

$$\text{Time} = (\text{B1} * \text{Viscosity}) / (\text{Weight} - \text{B2}) + \text{EB1B2E}$$

stormer

Viscosity

Wt

Time

studres

studres(object)

object

[residualsstdres](#)

summary.loglm

loglm

```
## S3 method for class 'loglm'  
summary(object, fitted = FALSE, ...)
```

```
object  
fitted          TRUEfitted = TRUE  
...
```

```
summary()"loglm"summary(x)xsummary.loglm(x)
```

print.summary.loglm

```
formula          object  
tests  
oe               fitted = TRUENULL
```

[loglmsummary](#)

summary.negbin

summary.glm

```
## S3 method for class 'negbin'  
summary(object, dispersion = 1, correlation = FALSE, ...)
```

```
object          negbinglmglm.nb  
dispersion      summary.glm  
correlation     summary.glm  
...
```

```
summary.glmsummary()"negbin"summary(x)xsummary.negbin(x)
```

```
summary.glm
```

```
summary.glm
```

```
summaryglm.nbnegative.binomialanova.negbin
```

```
summary(glm.nb(Days ~ Eth*Age*Lrn*Sex, quine, link = log))
```

```
summary.rlm
```

```
summary"rlm"
```

```
## S3 method for class 'rlm'  
summary(object, method = c("XtX", "XtWX"), correlation = FALSE, ...)
```

```
object          rlmrlm  
method  
correlation  
...
```

```
summary()"rlm"summary(x)xsummary.rlm(x)
```

```
summary  
correlation  
cov.unscaled  
sigma  
stddev  
df  
coefficients  
terms
```

summary

```
summary(rlm(calls ~ year, data = phones, maxit = 50))
```

survey

survey

Sex "Male""Female"
Wr.Hnd
NW.Hnd
W.Hnd "Left""Right"
Fold "R on L""L on R""Neither"
Pulse
Clap "Right""Left""Neither"
Exer "Freq""Some""None"
Smoke "Heavy""Regul""Occas""Never"
Height
M.I "Metric""Imperial"
Age

synth.tr

synth.trsynth.tesynth.trsynth.te

synth.tr
synth.te

xs
ys
yc

theta.md

theta

theta.md(y, mu, dfr, weights, limit = 20, eps = .Machine\$double.eps^0.25)

theta.ml(y, mu, n, weights, limit = 10, eps = .Machine\$double.eps^0.25,
trace = FALSE)

theta.mm(y, mu, dfr, weights, limit = 10, eps = .Machine\$double.eps^0.25)

y
mu
n weights
dfr theta
weights
limit
eps
trace

theta.md

theta.ml

theta.mmtheta $\sum (y - \mu)^2 / (\mu + \mu^2 / \theta)$

thetatheta.ml"SE"

glm.nb

```
quine.nb <- glm.nb(Days ~ .^2, data = quine)
theta.md(quine$Days, fitted(quine.nb), dfr = df.residual(quine.nb))
theta.ml(quine$Days, fitted(quine.nb))
theta.mm(quine$Days, fitted(quine.nb), dfr = df.residual(quine.nb))

## weighted example
yeast <- data.frame(cbind(numbers = 0:5, fr = c(213, 128, 37, 18, 3, 1)))
fit <- glm.nb(numbers ~ 1, weights = fr, data = yeast)
summary(fit)
mu <- fitted(fit)
theta.md(yeast$numbers, mu, dfr = 399, weights = yeast$fr)
theta.ml(yeast$numbers, mu, limit = 15, weights = yeast$fr)
theta.mm(yeast$numbers, mu, dfr = 399, weights = yeast$fr)
```

topo

topo

topo

x

y

z

Traffic

jj

Traffic

year
day
limit
y

truehist

```
truehist(data, nbins = "Scott", h, x0 = -h/1000,
          breaks, prob = TRUE, xlim = range(breaks),
          ymax = max(est), col = "cyan",
          xlab = deparse(substitute(data)), bty = "n", ...)
```

data	NA
nbins	"Scott""Freedman-Diaconis""FD"
h	nbins
x0	$x_0 + h * (... , -1, 0, 1, ...)$
breaks	hnbins
prob	
xlim	
ymax	
col	
xlab	data
bty	
...	rectplot

breakshhbreakshnbinsh

hist

ucv

ucv(x, nb = 1000, lower, upper)

x

nb

lowerupper

bcvwidth.SJdensity

ucv(geyser\$duration)

UScereal

UScereal

UScereal

mfr

calories

protein

fat

sodium

fibre

carbo

sugars

shelf

potassium

vitamins

<https://lib.stat.cmu.edu/datasets/1993.expo/>

UScrime

UScrime

M
So
Ed
Po1
Po2
LF
M.F
Pop
NW
U1
U2
GDP
Ineq
Prob
Time
y

VA

VA

stime
status
treat
age
Karn
diag.time
cell
prior

waders

waders

waders

S1

S2

S3

S4

S5

S6

S7

S8

S9

S10

S11

S12

S13

S14

S15

S16

S17

S18

S19

```
plot(corresp(waders, nf=2))
```

```
whiteside
```

```
whiteside
```

```
whiteside
```

```
Insul
```

```
Temp
```

```
Gas
```

```
require(lattice)
xyplot(Gas ~ Temp | Insul, whiteside, panel =
  function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.lmline(x, y, ...)
  }, xlab = "Average external temperature (deg. C)",
  ylab = "Gas consumption (1000 cubic feet)", aspect = "xy",
  strip = function(...) strip.default(..., style = 1))
```

```

gasB <- lm(Gas ~ Temp, whiteside, subset = Insul=="Before")
gasA <- update(gasB, subset = Insul=="After")
summary(gasB)
summary(gasA)
gasBA <- lm(Gas ~ Insul/Temp - 1, whiteside)
summary(gasBA)

gasQ <- lm(Gas ~ Insul/(Temp + I(Temp^2)) - 1, whiteside)
coef(summary(gasQ))

gasPR <- lm(Gas ~ Insul + Temp, whiteside)
anova(gasPR, gasBA)
options(contrasts = c("contr.treatment", "contr.poly"))
gasBA1 <- lm(Gas ~ Insul*Temp, whiteside)
coef(summary(gasBA1))

```

width.SJ

```
width.SJ(x, nb = 1000, lower, upper, method = c("ste", "dpi"))
```

```

x
nb
upperlower      method = "ste"
method          "ste""dpi"

```

$n \geq$ [bw.SJ](#)

[ucvbcvdensity](#)

```
width.SJ(geyser$duration, method = "dpi")
width.SJ(geyser$duration)
```

```
width.SJ(galaxies, method = "dpi")
width.SJ(galaxies)
```

`write.matrix`

```
write.matrix(x, file = "", sep = " ", blocksize)
```

```
x  
file      ""  
sep  
blocksize blocksize
```

```
xblocksize  
x
```

```
x
```

`write.table`

`wtloss`

`wtloss`

Days
Weight


```
## IGNORE_RDIFF_BEGIN
wtloss.fm <- nls(Weight ~ b0 + b1*2^(-Days/th),
  data = wtloss, start = list(b0=90, b1=95, th=120))
wtloss.fm
## IGNORE_RDIFF_END
plot(wtloss)
with(wtloss, lines(Days, fitted(wtloss.fm)))
```

Matrix

abIndex-class

```
"abIndex" classnumeric2:1000000c(0:1e5, 1000:1e6)
packageDescription("Matrix")$Maintainer
```

```
new("abIndex", ...)as(x, "abIndex")xabIseq()c(...)
```

```
kind character("int32", "double", "rleDiff")
x "numLike"0kind != "rleDiff"
rleD "rleDiff"rle
```

```
signature(x = "abIndex")
signature(x = "abIndex", i = "index", j = "ANY", drop = "ANY")
signature(from = "numeric", to = "abIndex")
signature(from = "abIndex", to = "numeric")
signature(from = "abIndex", to = "integer")
signature(x = "abIndex")
signature(e1 = "numeric", e2 = "abIndex")Ops
signature(e1 = "abIndex", e2 = "abIndex")
signature(e1 = "abIndex", e2 = "numeric")
signature(x = "abIndex")
("abIndex")showshow(<rleDiff>)
("abIndex")
("abIndex")
```

```
packageDescription("Matrix")$Maintainer
```

```
rlenumeric
```

```
showClass("abIndex")
ii <- c(-3:40, 20:70)
str(ai <- as(ii, "abIndex"))# note
ai # -> show() method

stopifnot(identical(-3:20,
                    as(abIseq1(-3,20), "vector")))
```

```
abIseq
```

```
"abIndex"
abIseq()seq"abIndex"
abIseq1()abIseq1(n,m)n:m
c(x, ...) "abIndex"x

abIseq1(from = 1, to = 1)
abIseq (from = 1, to = 1, by = ((to - from)/(length.out - 1)),
       length.out = NULL, along.with = NULL)

## S3 method for class 'abIndex'
c(...)

fromto
by
length.out      seqseq.int
along.with
...             "abIndex""abIndex"

"abIndex"

abIndexrep2abI()rle

stopifnot(identical(-3:20,
                    as(abIseq1(-3,20), "vector")))
```

```
try( ## (arithmetic) not yet implemented
abIseq(1, 50, by = 3)
)
```

all.equal-methods

`all.equal()`[Matrix](#)

`all.equal.numericas.vector()`
`"sparseVector"`[showMethods\("all.equal"\)](#)

`showMethods("all.equal")`

```
(A <- spMatrix(3,3, i= c(1:3,2:1), j=c(3:1,1:2), x = 1:5))
ex <- expand(lu. <- lu(A))
stopifnot( all.equal(as(A[lu.@p + 1L, lu.@q + 1L], "CsparseMatrix"),
                    lu.@L %*% lu.@U),
          with(ex, all.equal(as(P %*% A %*% t(Q), "CsparseMatrix"),
                              L %*% U)),
          with(ex, all.equal(as(A, "CsparseMatrix"),
                              t(P) %*% L %*% U %*% Q)))
```

asUniqueT

`TsparseMatrix`[\(i,j\)](#)

```
anyDuplicatedT(x, ...)
isUniqueT(x, byrow = FALSE, isT = is(x, "TsparseMatrix"))
asUniqueT(x, byrow = FALSE, isT = is(x, "TsparseMatrix"))
aggregateT(x)
```

x	<code>anyDuplicatedT</code> <code>aggregateT</code> <code>x</code> TsparseMatrix <code>asUniqueT</code> <code>x</code> Matrix <code>x</code> TsparseMatrix
...	anyDuplicated
byrow	x
isT	<code>x</code> TsparseMatrix

```
anyDuplicatedT(x)(i,j)x
isUniqueT(x)TRUExTsparseMatrix(i,j)FALSE
asUniqueT(x)TsparseMatrixx(i,j)(i,j)
aggregateT(x)
```

TsparseMatrix

```
example("dgTMatrix-class", echo=FALSE)
## -> 'T2' with (i,j,x) slots of length 5 each
T2u <- asUniqueT(T2)
stopifnot(## They "are" the same (and print the same):
          all.equal(T2, T2u, tol=0),
          ## but not internally:
          anyDuplicatedT(T2) == 2,
          anyDuplicatedT(T2u) == 0,
          length(T2 @x) == 5,
          length(T2u@x) == 3)

isUniqueT(T2) # FALSE
isUniqueT(T2u) # TRUE

T3 <- T2u
T3[1, c(1,3)] <- 10; T3[2, c(1,5)] <- 20
T3u <- asUniqueT(T3)
str(T3u) # sorted in 'j', and within j, sorted in i
stopifnot(isUniqueT(T3u))

## Logical l.TMatrix and n.TMatrix :
(L2 <- T2 > 0)
validObject(L2u <- asUniqueT(L2))
(N2 <- as(L2, "nMatrix"))
validObject(N2u <- asUniqueT(N2))
stopifnot(N2u@i == L2u@i, L2u@i == T2u@i, N2@i == L2@i, L2@i == T2@i,
          N2u@j == L2u@j, L2u@j == T2u@j, N2@j == L2@j, L2@j == T2@j)
# now with a nasty NA [partly failed in Matrix 1.1-5]:
L.0N <- L.1N <- L2
L.0N@x[1:2] <- c(FALSE, NA)
L.1N@x[1:2] <- c(TRUE, NA)
validObject(L.0N)
validObject(L.1N)
(m.0N <- as.matrix(L.0N))
(m.1N <- as.matrix(L.1N))
stopifnot(identical(10L, which(is.na(m.0N))), !anyNA(m.1N))
symnum(m.0N)
symnum(m.1N)
```

band-methods

triutrilband

```
band(x, k1, k2, ...)
triu(x, k = 0L, ...)
tril(x, k = 0L, ...)
```

```

x
kk1k2          k1 <= k2k = 0k
...

```

```

triu(x, k)band(x, k, dim(x)[2])tril(x, k)band(x, -dim(x)[1], k)

```

[triangularMatrix](#)[sparseMatrix](#)[x](#)

[matrix](#)

[bandSparse](#)

```

## A random sparse matrix :
set.seed(7)
m <- matrix(0, 5, 5)
m[sample(length(m), size = 14)] <- rep(1:9, length=14)
(mm <- as(m, "CsparseMatrix"))

tril(mm)          # lower triangle
tril(mm, -1)      # strict lower triangle
triu(mm, 1)       # strict upper triangle
band(mm, -1, 2)   # general band
(m5 <- Matrix(rnorm(25), ncol = 5))
tril(m5)          # lower triangle
tril(m5, -1)      # strict lower triangle
triu(m5, 1)       # strict upper triangle
band(m5, -1, 2)   # general band
(m65 <- Matrix(rnorm(30), ncol = 5)) # not square
triu(m65)         # result not "dtrMatrix" unless square
(sm5 <- crossprod(m65)) # symmetric
  band(sm5, -1, 1) # "dsyMatrix": symmetric band preserves symmetry property
as(band(sm5, -1, 1), "sparseMatrix") # often preferable
(sm <- round(crossprod(triu(mm/2)))) # sparse symmetric ("dsC*")
band(sm, -1,1) # remains "dsC", *however*
band(sm, -2,1) # -> "dgC"

```

bandSparse

```
bandSparse(n, m = n, k, diagonals, symmetric = FALSE,
           repr = "C", giveCsparse = (repr == "C"))
```

```
nm          (n, m) = (nrow, ncol)
k           band(*, k)k=0
diagonals   nMatrix
            diagonals  $n' \times dd <- \text{length}(k)n' \geq \min(n, m)$ diagonals
symmetric   symmetricMatrixkdiagonals
repr        character"C""T""R"CsparseMatrixTsparseMatrixRsparseMatrix
giveCsparse reprCsparseMatrixTsparseMatrixTRUErepr
```

```
classCsparseMatrix  $n \times m$ 
```

```
bandbdiagdiagsparseMatrixMatrix
```

```
diags <- list(1:30, 10*(1:20), 100*(1:20))
s1 <- bandSparse(13, k = -c(0:2, 6), diag = c(diags, diags[2]), symm=TRUE)
s1
s2 <- bandSparse(13, k = c(0:2, 6), diag = c(diags, diags[2]), symm=TRUE)
stopifnot(identical(s1, t(s2)), is(s1, "dsCMatrix"))

## a pattern Matrix of *full* (sub-)diagonals:
bk <- c(0:4, 7, 9)
(s3 <- bandSparse(30, k = bk, symm = TRUE))

## If you want a pattern matrix, but with "sparse"-diagonals,
## you currently need to go via logical sparse:
llis <- lapply(list(rpois(20, 2), rpois(20, 1), rpois(20, 3))[c(1:3, 2:3, 3:2)],
              as.logical)
(s4 <- bandSparse(20, k = bk, symm = TRUE, diag = llis))
(s4. <- as(drop0(s4), "nsparseMatrix"))

n <- 1e4
bk <- c(0:5, 7, 11)
bMat <- matrix(1:8, n, 8, byrow=TRUE)
bLis <- as.data.frame(bMat)
B <- bandSparse(n, k = bk, diag = bLis)
Bs <- bandSparse(n, k = bk, diag = bLis, symmetric=TRUE)
B [1:15, 1:30]
```

```

Bs[1:15, 1:30]
## can use a list *or* a matrix for specifying the diagonals:
stopifnot(identical(B, bandSparse(n, k = bk, diag = bMat)),
  identical(Bs, bandSparse(n, k = bk, diag = bMat, symmetric=TRUE))
    , inherits(B, "dtCMatrix") # triangular!
)

```

bdiag

```

bdiag(...)
.bdiag(lst)

```

```

...      list
lst      list

```

```

bdiag().bdiag()

```

```

bdiag()CsparseMatrix.bdiag()TsparseMatrix

```

```

 $k \times k$  bdiag_m()

```

```

TsparseMatrix

```

```

DiagonaldiagonalMatrixkronecker"Matrix"
bandSparse
bdiag()

```

```

bdiag(matrix(1:4, 2), diag(3))
## combine "Matrix" class and traditional matrices:
bdiag(Diagonal(2), matrix(1:3, 3,4), diag(3:2))

mlist <- list(1, 2:3, diag(x=5:3), 27, cbind(1,3:6), 100:101)
bdiag(mlist)
stopifnot(identical(bdiag(mlist),
  bdiag(lapply(mlist, as.matrix))))

```



```

ml <- c(as(matrix((1:24)%% 11 == 0, 6,4),"nMatrix"),
      rep(list(Diagonal(2, x=TRUE)), 3))
mln <- c(ml, Diagonal(x = 1:3))
stopifnot(is(bdiag(ml), "lsparseMatrix"),
          is(bdiag(mln),"dsparseMatrix") )

## random (diagonal-)block-triangular matrices:
rblockTri <- function(nb, max.ni, lambda = 3) {
  .bdiag(replicate(nb, {
    n <- sample.int(max.ni, 1)
    tril(Matrix(rpois(n * n, lambda = lambda), n, n)) )))
}

(T4 <- rblockTri(4, 10, lambda = 1))
image(T1 <- rblockTri(12, 20))

##' Fast version of Matrix :: .bdiag() -- for the case of *many* (k x k) matrices:
##' @param lmat list(<mat1>, <mat2>, ....., <mat_N>) where each mat_j is a k x k 'matrix'
##' @return a sparse (N*k x N*k) matrix of class \code{"\linkS4class{dgCMatrix}"}.
bdiag_m <- function(lmat) {
  ## Copyright (C) 2016 Martin Maechler, ETH Zurich
  if(!length(lmat)) return(new("dgCMatrix"))
  stopifnot(is.list(lmat), is.matrix(lmat[[1]]),
            (k <- (d <- dim(lmat[[1]]))[1]) == d[2], # k x k
            all(vapply(lmat, dim, integer(2)) == k)) # all of them
  N <- length(lmat)
  if(N * k > .Machine$integer.max)
    stop("resulting matrix too large; would be M x M, with M=", N*k)
  M <- as.integer(N * k)
  ## result: an M x M matrix
  new("dgCMatrix", Dim = c(M,M),
      ## 'i :' maybe there's a faster way (w/o matrix indexing), but elegant?
      i = as.vector(matrix(0L:(M-1L), nrow=k)[, rep(seq_len(N), each=k)]),
      p = k * 0L:M,
      x = as.double(unlist(lmat, recursive=FALSE, use.names=FALSE)))
}

l12 <- replicate(12, matrix(rpois(16, lambda = 6.4), 4, 4),
                      simplify=FALSE)
dim(T12 <- bdiag_m(l12))# 48 x 48
T12[1:20, 1:20]

```

```
boolmatmult-methods    %&%
```

```

nMatrix
%&%matrix

```

```
"nMatrix""ldiMatrix"
```

sparseVector

```
signature(x = "ANY", y = "ANY")
signature(x = "ANY", y = "Matrix")
signature(x = "Matrix", y = "ANY")
signature(x = "nMatrix", y = "nMatrix")
signature(x = "nMatrix", y = "nsparseMatrix")
signature(x = "nsparseMatrix", y = "nMatrix")
signature(x = "nsparseMatrix", y = "nsparseMatrix")
signature(x = "sparseVector", y = "sparseVector")
```

```
0M@x"dMatrix""lMatrix"drop0(M)
MM != 0M != FALSE
xy $\text{nsparseMatrix}_{m_i j}$ 
```

%%crossprod()tcrossprod()

```
set.seed(7)
L <- Matrix(rnorm(20) > 1, 4, 5)
(N <- as(L, "nMatrix"))
L. <- L; L.[1:2,1] <- TRUE; L.@x[1:2] <- FALSE; L. # has "zeros" to drop0()
D <- Matrix(round(rnorm(30)), 5, 6) # -> values in -1:1 (for this seed)
L %%% D
stopifnot(identical(L %%% D, N %%% D),
  all(L %%% D == as((L %*% abs(D)) > 0, "sparseMatrix")))

## cross products , possibly with boolArith = TRUE :
crossprod(N) # -> sparse pattern 'n' (TRUE/FALSE : boolean arithmetic)
crossprod(N + 0) # -> numeric Matrix (with same "pattern")
stopifnot(all(crossprod(N) == t(N) %%% N),
  identical(crossprod(N), crossprod(N + 0, boolArith=TRUE)),
  identical(crossprod(L), crossprod(N, boolArith=FALSE)))
crossprod(D, boolArith = TRUE) # pattern: "nsCMatrix"
crossprod(L, boolArith = TRUE) # ditto
crossprod(L, boolArith = FALSE) # numeric: "dsCMatrix"
```

BunchKaufman-class

BunchKaufmanpBunchKaufmann $\times nA$

$$A = UD_U U' = LD_L L'$$

$$D_U D_L b_U b_L 1 \times 12 \times 2U = \prod_{k=1}^{b_U} P_k U_k b_U L = \prod_{k=1}^{b_L} P_k L_k b_L$$
$$2b_U + 12b_L + 1nn\text{BunchKaufmann}(n+1)/2p\text{BunchKaufman}$$

```

DimDimnames MatrixFactorization
uplo "U""L"x
x n*nBunchKaufmann*(n+1)/2pBunchKaufmann=Dim[1]dsytrfdsptrf
perm n=Dim[1]dsytrfdsptrf

```

BunchKaufmanFactorizationMatrixFactorizationBunchKaufmanFactorization

```

new("BunchKaufman", ...)new("pBunchKaufman", ...)BunchKaufman(x)xdsyMatrix
dspMatrix

```

```

coerce signature(from = "BunchKaufman", to = "dtrMatrix")dtrMatrix
coerce signature(from = "pBunchKaufman", to = "dtpMatrix")dtpMatrix
determinant signature(from = "p?BunchKaufman", logarithm = "logical")A
expand1 signature(x = "p?BunchKaufman")expand1-methods
expand2 signature(x = "p?BunchKaufman")expand2-methods
solve signature(a = "p?BunchKaufman", b = .)solve-methods

```

```

< 1.6-0BunchKaufmandtrMatrixpBunchKaufmandtpMatrixn(n + 1)/2n × n1.6-0dtrMatrix
dtpMatrixBunchKaufmanpBunchKaufman
as(., "dtrMatrix")as(., "dtpMatrix")

```

<https://netlib.org/lapack/double/dsytrf.f><https://netlib.org/lapack/double/dsptrf.f>

dsyMatrix
BunchKaufmanexpand1expand2

```

showClass("BunchKaufman")
set.seed(1)

n <- 6L
(A <- forceSymmetric(Matrix(rnorm(n * n), n, n)))

## With dimnames, to see that they are propagated :
dimnames(A) <- rep.int(list(paste0("x", seq_len(n))), 2L)

(bk.A <- BunchKaufman(A))
str(e.bk.A <- expand2(bk.A, complete = FALSE), max.level = 2L)
str(E.bk.A <- expand2(bk.A, complete = TRUE), max.level = 2L)

```

```

## Underlying LAPACK representation
(m.bk.A <- as(bk.A, "dtrMatrix"))
stopifnot(identical(as(m.bk.A, "matrix"), `dim<-`(bk.A@x, bk.A@Dim)))

## Number of factors is 2*b+1, b <= n, which can be nontrivial ...
(b <- (length(E.bk.A) - 1L) %% 2L)

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

## A ~ U DU U', U := prod(Pk Uk) in floating point
stopifnot(exprs = {
  identical(names(e.bk.A), c("U", "DU", "U."))
  identical(e.bk.A[["U" ]], Reduce(`%*%`, E.bk.A[seq_len(b)]))
  identical(e.bk.A[["U." ]], t(e.bk.A[["U" ]]))
  ae1(A, with(e.bk.A, U %*% DU %*% U.))
})

## Factorization handled as factorized matrix
b <- rnorm(n)
stopifnot(identical(det(A), det(bk.A)),
  identical(solve(A, b), solve(bk.A, b)))

```

BunchKaufman-methods

$n \times nA$

$$A = UD_U U' = LD_L L'$$

$$D_U D_L b_U b_L 1 \times 12 \times 2U = \prod_{k=1}^{b_U} P_k U_k b_U L = \prod_{k=1}^{b_L} P_k L_k b_L$$

dsytrfdsptrf

```

BunchKaufman(x, ...)
## S4 method for signature 'dsyMatrix'
BunchKaufman(x, warnSing = TRUE, ...)
## S4 method for signature 'dspMatrix'
BunchKaufman(x, warnSing = TRUE, ...)
## S4 method for signature 'matrix'
BunchKaufman(x, uplo = "U", ...)

```

```

x          Matrixxuplo
warnSing   x
uplo       "U" "L" x
...

```

[BunchKaufmanFactorization](#)[BunchKaufmanpackedMatrixp](#)[BunchKaufman](#)

<https://netlib.org/lapack/double/dsytrf.f>
<https://netlib.org/lapack/double/dsptrf.f>

[BunchKaufman](#)[BunchKaufman](#)

[dsyMatrix](#)[dspMatrix](#)

[expand1](#)[expand2](#)

[Cholesky](#)[Schur](#)[luqr](#)

```
showMethods("BunchKaufman", inherited = FALSE)
set.seed(0)

data(CAex, package = "Matrix")
class(CAex) # dgCMatrix
isSymmetric(CAex) # symmetric, but not formally

A <- as(CAex, "symmetricMatrix")
class(A) # dsCMatrix

## Have methods for denseMatrix (unpacked and packed),
## but not yet sparseMatrix ...
## Not run:
(bk.A <- BunchKaufman(A))

## End(Not run)
(bk.A <- BunchKaufman(as(A, "unpackedMatrix"))))

## A ~ U D U' in floating point
str(e.bk.A <- expand2(bk.A), max.level = 2L)
stopifnot(all.equal(as(A, "matrix"), as(Reduce(`%*%`, e.bk.A), "matrix")))
```

CAex

[eigen\(\)](#)<https://stat.ethz.ch/mailman/listinfo/r-help>

`data(CAex)`

72×72 [dgCMatrix](#)

[eigen](#)(CAex)[eigen](#)(CAex, EISPACK=TRUE)

```

data(CAex, package = "Matrix")
str(CAex) # of class "dgCMatrix"

image(CAex)# -> it's a simple band matrix with 5 bands
## and the eigen values are basically 1 (42 times) and 0 (30 x):
zapsmall(ev <- eigen(CAex, only.values=TRUE)$values)
## i.e., the matrix is symmetric, hence
sCA <- as(CAex, "symmetricMatrix")
## and
stopifnot(class(sCA) == "dsCMatrix",
           as(sCA, "matrix") == as(CAex, "matrix"))

```

cbind2-methods

```

cbindrbind...cbind()rbind()cbind2rbind2
cbind2()rbind2()'Matrix'

## cbind(..., deparse.level = 1)
## rbind(..., deparse.level = 1)

## S4 method for signature 'Matrix,Matrix'
cbind2(x, y, ...)
## S4 method for signature 'Matrix,Matrix'
rbind2(x, y, ...)

...          [cr]bind[cr]bind2cbindcbind2
deparse.level  cbind
xy

class...
sparseMatrixsparseMatrix()

cbindcbind2
cbind2()rbind2()"denseMatrix""diagonalMatrix""indMatrix"

(a <- matrix(c(2:1,1:2), 2,2))

(M1 <- cbind(0, rbind(a, 7))) # a traditional matrix

D <- Diagonal(2)
(M2 <- cbind(4, a, D, -1, D, 0)) # a sparse Matrix

stopifnot(validObject(M2), inherits(M2, "sparseMatrix"),
           dim(M2) == c(2,9))

```

CHMfactor-class

CHMfactor $n \times n$ A

$$P_1 A P_1' = L_1 D L_1' \stackrel{D_{jj} \geq 0}{=} L L'$$

$$A = P_1' L_1 D L_1' P_1 \stackrel{D_{jj} \geq 0}{=} P_1' L L' P_1$$

$$P_1 L_1 D L = L_1 \sqrt{D} A D \sqrt{D}$$

CHMfactorcholmod_factor_structCHMsimplCHMsimpl[dn]CHMsimpl[dn]CHMsimpl[dn]

isLDL(x)

x CHMfactorCholesky

isLDL(x)TRUEFALSETRUExL₁ − I + DFALSExL

CHMfactor

DimDimnames MatrixFactorization

colcount Dim[1]

perm Dim[1]perm

type orderingis_llis_superis_monotonicmaxcsizemaxsizecholmod_factor_struct
is_supermaxcsizemaxsizeis_llis_monotonic

CHMsimplnCHMsimpl

nz Dim[1]

p Dim[1]+1jip[j]+seq_len(nz[j]))+1

i sum(nz)ijnz[j]

prvnxt Dim[1]+2ixj <- Dim[1]+2j <- nxt[j+1]+1nxt[j+1] = -1j <- Dim[1]+1j <-
prv[j+1]+1prv[j+1] = -1

dCHMsimpl

x iLtype[2]L₁ − I + D

CHMsimpl

superpipx nsuper+1nsupersuper[j]+1jjs[pi[j]+seq_len(pi[j+1]-pi[j]))+1j
x[px[j]+seq_len(px[j+1]-px[j]))+1x

s Dim[1]s

dCHMsimpl

x prod(Dim)

MatrixFactorization

`new("dCHMsimpl", ...)dCHMsimpldCHMSuperCholesky(x, ...)xsparseMatrixdsCMatrix
newnCHMsimplnCHMSuper`

`coerce signature(from = "CHMsimpl", to = "dtCMatrix")dtCMatrix $LL_1 - I + D$
from@type[2]`

`coerce signature(from = "CHMSuper", to = "dgCMatrix")dgCMatrixLdgCMatrix
dtCMatrix`

`determinant signature(from = "CHMfactor", logarithm = "logical")sqrtsqrt
= FALSEAsqrt = TRUE $L = L_1\sqrt{D}$ NaN $D\sqrt{D}$ TRUEsqrtTRUE< 1.6-0sqrt
options(Matrix.warnSqrtDefault = 0)`

`diag signature(x = "CHMfactor")nDL`

`expand signature(x = "CHMfactor")expand-methods`

`expand1 signature(x = "CHMsimpl")expand1-methods`

`expand1 signature(x = "CHMSuper")expand1-methods`

`expand2 signature(x = "CHMsimpl")expand2-methods`

`expand2 signature(x = "CHMSuper")expand2-methods`

`image signature(x = "CHMfactor")image-methods`

`nnzero signature(x = "CHMfactor")nnzero-methods`

`solve signature(a = "CHMfactor", b = .)solve-methods`

`update signature(object = "CHMfactor")objectparentmultparentdsCMatrixdgCMatrix
mult
F(parent) + mult[1] * IF(parent)mult[1]F = identityparentF = tcrossprodparent
F(parent)Sobject = Cholesky(S, ...)`

`updown signature(update = ., C = ., object = "CHMfactor")updown-methods`

<https://github.com/DrTimothyAldenDavis/SuiteSparseCHOLMOD/Include/cholmod.h>
cholmod_factor_struct

`dsCMatrix`

`Choleskyupdownexpand1expand2`


```

showClass("dCHMsimpl")
showClass("dCHMsuper")
set.seed(2)

m <- 1000L
n <- 200L
M <- rsparsematrix(m, n, 0.01)
A <- crossprod(M)

## With dimnames, to see that they are propagated :
dimnames(A) <- dn <- rep.int(list(paste0("x", seq_len(n))), 2L)

(ch.A <- Cholesky(A)) # pivoted, by default
str(e.ch.A <- expand2(ch.A, LDL = TRUE), max.level = 2L)
str(E.ch.A <- expand2(ch.A, LDL = FALSE), max.level = 2L)

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

## A ~ P1' L1 D L1' P1 ~ P1' L L' P1 in floating point
stopifnot(exprs = {
  identical(names(e.ch.A), c("P1.", "L1", "D", "L1.", "P1"))
  identical(names(E.ch.A), c("P1.", "L", "L.", "P1"))
  identical(e.ch.A[["P1"]],
    new("pMatrix", Dim = c(n, n), Dimnames = c(list(NULL), dn[2L]),
      margin = 2L, perm = invertPerm(ch.A@perm, 0L, 1L)))
  identical(e.ch.A[["P1."]], t(e.ch.A[["P1"]]))
  identical(e.ch.A[["L1."]], t(e.ch.A[["L1"]]))
  identical(E.ch.A[["L." ]], t(E.ch.A[["L" ]]))
  identical(e.ch.A[["D"]], Diagonal(x = diag(ch.A)))
  all.equal(E.ch.A[["L"]], with(e.ch.A, L1 %%% sqrt(D)))
  ae1(A, with(e.ch.A, P1. %%% L1 %%% D %%% L1. %%% P1))
  ae1(A, with(E.ch.A, P1. %%% L %%% L. %%% P1))
  ae2(A[ch.A@perm + 1L, ch.A@perm + 1L], with(e.ch.A, L1 %%% D %%% L1.))
  ae2(A[ch.A@perm + 1L, ch.A@perm + 1L], with(E.ch.A, L %%% L. ))
})

## Factorization handled as factorized matrix
## (in some cases only optionally, depending on arguments)
b <- rnorm(n)
stopifnot(identical(det(A), det(ch.A, sqrt = FALSE)),
  identical(solve(A, b), solve(ch.A, b, system = "A")))

u1 <- update(ch.A, A, mult = sqrt(2))
u2 <- update(ch.A, t(M), mult = sqrt(2)) # updating with crossprod(M), not M
stopifnot(all.equal(u1, u2, tolerance = 1e-14))

```

chol-methods

$n \times nAL'$

$$P_1AP'_1 = LL'$$

$$A = P_1' L L' P_1$$

P_1

`denseMatrix``dpstrfdpotrfdpptrf` P_1

`sparseMatrix``cholmod_analyzecholmod_factorize_p`

```
chol(x, ...)
## S4 method for signature 'dsyMatrix'
chol(x, pivot = FALSE, tol = -1, ...)
## S4 method for signature 'dspMatrix'
chol(x, ...)
## S4 method for signature 'dsCMatrix'
chol(x, pivot = FALSE, ...)
## S4 method for signature 'ddiMatrix'
chol(x, ...)
## S4 method for signature 'generalMatrix'
chol(x, uplo = "U", ...)
## S4 method for signature 'triangularMatrix'
chol(x, uplo = "U", ...)
```

```
x          Matrixxuploxpivot = FALSEx
pivot      xx
tol        pivot = TRUEtoltolnrow(x) * .Machine$double.eps * max(diag(x))
uplo       "U""L"x"U"xchol
...
```

`x``diagonalMatrix`

`x``chol(x, pivot = value)``Cholesky(x, perm = value, ...)` $P_1 L'$ `Cholesky``chol(x, pivot = TRUE)` $L' P_1$

`triangularMatrix``diagonalMatrix` L' `xxx`

<https://netlib.org/lapack/double/dpstrf.f><https://netlib.org/lapack/double/dpotrf.f><https://netlib.org/lapack/double/dpptrf.f>

<https://github.com/DrTimothyAldenDavis/SuiteSparse>`CHOLMOD/Include/cholmod.h`
`cholmod_factor_struct`

`chol``x`

`Cholesky` L'

```

showMethods("chol", inherited = FALSE)
set.seed(0)

## ---- Dense -----

## chol(x, pivot = value) wrapping Cholesky(x, perm = value)
selectMethod("chol", "dsyMatrix")

## Except in packed cases where pivoting is not yet available
selectMethod("chol", "dspMatrix")

## .... Positive definite .....

(A1 <- new("dsyMatrix", Dim = c(2L, 2L), x = c(1, 2, 2, 5)))
(R1.nopivot <- chol(A1))
(R1 <- chol(A1, pivot = TRUE))

## In 2-by-2 cases, we know that the permutation is 1:2 or 2:1,
## even if in general 'chol' does not say ...

stopifnot(exprs = {
  all.equal( A1, as(crossprod(R1.nopivot), "dsyMatrix"))
  all.equal(t(A1[2:1, 2:1]), as(crossprod(R1), "dsyMatrix"))
  identical(Cholesky(A1)@perm, 2:1) # because 5 > 1
})

## .... Positive semidefinite but not positive definite .....

(A2 <- new("dpoMatrix", Dim = c(2L, 2L), x = c(1, 2, 2, 4)))
try(R2.nopivot <- chol(A2)) # fails as not positive definite
(R2 <- chol(A2, pivot = TRUE)) # returns, with a warning and ...

stopifnot(exprs = {
  all.equal(t(A2[2:1, 2:1]), as(crossprod(R2), "dsyMatrix"))
  identical(Cholesky(A2)@perm, 2:1) # because 4 > 1
})

## .... Not positive semidefinite .....

(A3 <- new("dsyMatrix", Dim = c(2L, 2L), x = c(1, 2, 2, 3)))
try(R3.nopivot <- chol(A3)) # fails as not positive definite
(R3 <- chol(A3, pivot = TRUE)) # returns, with a warning and ...

## _Not_ equal: see details and examples in help("Cholesky")
all.equal(t(A3[2:1, 2:1]), as(crossprod(R3), "dsyMatrix"))

## ---- Sparse -----

## chol(x, pivot = value) wrapping
## Cholesky(x, perm = value, LDL = FALSE, super = FALSE)
selectMethod("chol", "dsCMatrix")

## Except in diagonal cases which are handled "directly"
selectMethod("chol", "ddiMatrix")

(A4 <- toeplitz(as(c(10, 0, 1, 0, 3), "sparseVector"))))

```

```

(ch.A4.nopivot <- Cholesky(A4, perm = FALSE, LDL = FALSE, super = FALSE))
(ch.A4 <- Cholesky(A4, perm = TRUE, LDL = FALSE, super = FALSE))
(R4.nopivot <- chol(A4))
(R4 <- chol(A4, pivot = TRUE))

det4 <- det(A4)
b4 <- rnorm(5L)
x4 <- solve(A4, b4)

stopifnot(exprs = {
  identical(R4.nopivot, expand1(ch.A4.nopivot, "L."))
  identical(R4, expand1(ch.A4, "L."))
  all.equal(A4, crossprod(R4.nopivot))
  all.equal(A4[ch.A4@perm + 1L, ch.A4@perm + 1L], crossprod(R4))
  all.equal(diag(R4.nopivot), sqrt(diag(ch.A4.nopivot)))
  all.equal(diag(R4), sqrt(diag(ch.A4)))
  all.equal(sqrt(det4), det(R4.nopivot))
  all.equal(sqrt(det4), det(R4))
  all.equal(det4, det(ch.A4.nopivot, sqrt = FALSE))
  all.equal(det4, det(ch.A4, sqrt = FALSE))
  all.equal(x4, solve(R4.nopivot, solve(t(R4.nopivot), b4)))
  all.equal(x4, solve(ch.A4.nopivot, b4))
  all.equal(x4, solve(ch.A4, b4))
})

```

chol2inv-methods

$$UL(U'U)^{-1}(LL')^{-1}$$

$$(X'X)^{-1}RXXR$$

```

chol2inv(x, ...)
## S4 method for signature 'dtrMatrix'
chol2inv(x, ...)
## S4 method for signature 'dtCMatrix'
chol2inv(x, ...)
## S4 method for signature 'generalMatrix'
chol2inv(x, uplo = "U", ...)

```

```

x          MatrixcholxuploxMatrix
uplo       "U""L""X""U"chol2inv
...

```

```

symmetricMatrixdiagonalMatrixxxxx

```

```

chol2invx
cholL'
solve

```

```

(A <- Matrix(cbind(c(1, 1, 1), c(1, 2, 4), c(1, 4, 16))))
(R <- chol(A))
(L <- t(R))
(R2i <- chol2inv(R))
(L2i <- chol2inv(R))
stopifnot(exprs = {
  all.equal(R2i, tcrossprod(solve(R)))
  all.equal(L2i, crossprod(solve(L)))
  all.equal(as(R2i %**% A, "matrix"), diag(3L)) # the identity
  all.equal(as(L2i %**% A, "matrix"), diag(3L)) # ditto
})

```

Cholesky-class

CholeskypCholeskyn $\times nA$

$$P_1 A P_1' = L_1 D L_1' = L L'$$

$$A = P_1' L_1 D L_1' P_1 = P_1' L L' P_1$$

$$P_1 L_1 D L = L_1 \sqrt{D}$$

$$L L' n n \text{Choleskyn} (n+1) / 2 p \text{Cholesky}$$

DimDimnames [MatrixFactorization](#)

uplo "U" "L" $\times L' L$

x $n \times n \text{Choleskyn} * (n+1) / 2 p \text{Choleskyn} = \text{Dim}[1] L L'$

perm Dim[1]perm

[CholeskyFactorizationMatrixFactorizationCholeskyFactorization](#)

new("Cholesky", ...)new("pCholesky", ...)Cholesky(x)xdsyMatrixdspMatrix
dpoMatrixdppMatrix

coerce signature(from = "Cholesky", to = "dtrMatrix")[dtrMatrixLL'](#)

coerce signature(from = "pCholesky", to = "dtpMatrix")[dtpMatrixLL'](#)

determinant signature(from = "p?Cholesky", logarithm = "logical")A

diag signature(x = "p?Cholesky") $n D L$

expand1 signature(x = "p?Cholesky")[expand1-methods](#)

expand2 signature(x = "p?Cholesky")[expand2-methods](#)

solve signature(a = "p?Cholesky", b = .)[solve-methods](#)

```
< 1.6-0CholeskydtrMatrixpCholeskydtpMatrixL1.6-0dtrMatrixdtpMatrixCholesky
pCholesky
as(., "dtrMatrix")as(., "dtpMatrix")
```

<https://netlib.org/lapack/double/dpstrf.f><https://netlib.org/lapack/double/dpotrf.f><https://netlib.org/lapack/double/dpptrf.f>
<https://www.netlib.org/lapack/lawnspdf/lawn161.pdf>

CHMfactor

dpoMatrixdppMatrix

Choleskyexpand1expand2

```
showClass("Cholesky")
set.seed(1)

m <- 30L
n <- 6L
(A <- crossprod(Matrix(rnorm(m * n), m, n)))

## With dimnames, to see that they are propagated :
dimnames(A) <- dn <- rep.int(list(paste0("x", seq_len(n))), 2L)

(ch.A <- Cholesky(A)) # pivoted, by default
str(e.ch.A <- expand2(ch.A, LDL = TRUE), max.level = 2L)
str(E.ch.A <- expand2(ch.A, LDL = FALSE), max.level = 2L)

## Underlying LAPACK representation
(m.ch.A <- as(ch.A, "dtrMatrix")) # which is L', not L, because
A@uplo == "U"
stopifnot(identical(as(m.ch.A, "matrix"), `dim<-`(ch.A@x, ch.A@Dim)))

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

## A ~ P1' L1 D L1' P1 ~ P1' L L' P1 in floating point
stopifnot(exprs = {
  identical(names(e.ch.A), c("P1.", "L1", "D", "L1.", "P1"))
  identical(names(E.ch.A), c("P1.", "L", "L.", "P1"))
  identical(e.ch.A[["P1"]],
    new("pMatrix", Dim = c(n, n), Dimnames = c(list(NULL), dn[2L]),
      margin = 2L, perm = invertPerm(ch.A@perm)))
  identical(e.ch.A[["P1."]], t(e.ch.A[["P1"]]))
  identical(e.ch.A[["L1."]], t(e.ch.A[["L1"]]))
  identical(E.ch.A[["L." ]], t(E.ch.A[["L" ]]))
  identical(e.ch.A[["D"]], Diagonal(x = diag(ch.A)))
  all.equal(E.ch.A[["L"]], with(e.ch.A, L1 %*% sqrt(D)))
  ae1(A, with(e.ch.A, P1. %*% L1 %*% D %*% L1. %*% P1))
})
```

```

ae1(A, with(E.ch.A, P1. %*% L %*% L. %*% P1))
ae2(A[ch.A@perm, ch.A@perm], with(e.ch.A, L1 %*% D %*% L1.))
ae2(A[ch.A@perm, ch.A@perm], with(E.ch.A, L %*% L. ))
})

## Factorization handled as factorized matrix
b <- rnorm(n)
all.equal(det(A), det(ch.A), tolerance = 0)
all.equal(solve(A, b), solve(ch.A, b), tolerance = 0)

## For identical results, we need the _unpivoted_ factorization
## computed by det(A) and solve(A, b)
(ch.A.nopivot <- Cholesky(A, perm = FALSE))
stopifnot(identical(det(A), det(ch.A.nopivot)),
           identical(solve(A, b), solve(ch.A.nopivot, b)))

```

Cholesky-methods

$n \times n$ A

$$P_1 A P_1' = L_1 D L_1' \stackrel{D_{jj} \geq 0}{=} LL'$$

$$A = P_1' L_1 D L_1' P_1 \stackrel{D_{jj} \geq 0}{=} P_1' L L' P_1$$

$$P_1 L_1 D L = L_1 \sqrt{D} A D \sqrt{D}$$

`denseMatrix` `dpstrfd` `potrfd` `pptrf` `P1`

`sparseMatrix` `cholmod_analyze` `cholmod_factorize_p`

```

Cholesky(A, ...)
## S4 method for signature 'dsyMatrix'
Cholesky(A, perm = TRUE, tol = -1, ...)
## S4 method for signature 'dspMatrix'
Cholesky(A, ...)
## S4 method for signature 'dsCMatrix'
Cholesky(A, perm = TRUE, LDL = !super, super = FALSE,
         Imult = 0, ...)
## S4 method for signature 'ddiMatrix'
Cholesky(A, ...)
## S4 method for signature 'generalMatrix'
Cholesky(A, uplo = "U", ...)
## S4 method for signature 'triangularMatrix'
Cholesky(A, uplo = "U", ...)
## S4 method for signature 'matrix'
Cholesky(A, uplo = "U", ...)

```

```

A           MatrixAuploAperm = FALSEperm = TRUEALDL = TRUELDL = FALSEA
perm        Aperm = FALSE
tol         perm = TRUEetoltolnrow(A) * .Machine$double.eps * max(diag(A))

```

```

LDL       $P_1' L_1 D L_1' P_1 L_1 - I + D P_1' L L' P_1 L = L_1 \sqrt{D}$ super = TRUEsuper = NALDL
          = TRUE
super    super = NA
Imult    A + Imult * diag(nrow(A))AImultAImult > max(rowSums(abs(A)) -
          diag(abs(A)))A + Imult * diag(nrow(A))
uplo     "U""L"A"U"chol
...

```

```

CholeskyCholeskyFactorizationMatrixcholCholeskyL'triangularMatrix
CholeskyFactorizationexpand1(x, "L")x
Cholesky(A, perm = TRUE)AAAtolP'LL'PAA

```

$$\frac{\|A - P'LL'P\|}{\|A\|}.$$

```

CholeskyFactorizationACholeskyAunpackedMatrixpackedMatrixsparseMatrixCholesky
pCholeskydCHMsimpldCHMsuper

```

```

https://netlib.org/lapack/double/dpstrf.fhttps://netlib.org/lapack/double/dpotrf.fhttps://netlib.org/lapack/double/dpptrf.f
https://github.com/DrTimothyAldenDavis/SuiteSparseCHOLMOD/Include/cholmod.h
cholmod_factor_struct
https://www.netlib.org/lapack/lawnspdf/lawn161.pdf

```

```

CholeskypCholeskydCHMsimpldCHMsuper
dpoMatrixdppMatrixdsCMatrix
cholL'Matrix
expand1expand2
BunchKaufmanSchurluqr

```

```

showMethods("Cholesky", inherited = FALSE)
set.seed(0)

```

```

## ---- Dense -----
## .... Positive definite .....

n <- 6L
(A1 <- crossprod(Matrix(rnorm(n * n), n, n)))

```



```

(ch.A1.nopivot <- Cholesky(A1, perm = FALSE))
(ch.A1 <- Cholesky(A1))
stopifnot(exprs = {
  length(ch.A1@perm) == ncol(A1)
  isPerm(ch.A1@perm)
  is.unsorted(ch.A1@perm) # typically not the identity permutation
  length(ch.A1.nopivot@perm) == 0L
})

## A ~ P1' L D L' P1 ~ P1' L L' P1 in floating point
str(e.ch.A1 <- expand2(ch.A1, LDL = TRUE), max.level = 2L)
str(E.ch.A1 <- expand2(ch.A1, LDL = FALSE), max.level = 2L)
stopifnot(exprs = {
  all.equal(as(A1, "matrix"), as(Reduce(`%*%`, e.ch.A1), "matrix"))
  all.equal(as(A1, "matrix"), as(Reduce(`%*%`, E.ch.A1), "matrix"))
})

## .... Positive semidefinite but not positive definite .....

A2 <- A1
A2[1L, ] <- A2[, 1L] <- 0
A2
try(Cholesky(A2, perm = FALSE)) # fails as not positive definite
ch.A2 <- Cholesky(A2) # returns, with a warning and ...
A2.hat <- Reduce(`%*%`, expand2(ch.A2, LDL = FALSE))
norm(A2 - A2.hat, "2") / norm(A2, "2") # 7.670858e-17

## .... Not positive semidefinite .....

A3 <- A1
A3[1L, ] <- A3[, 1L] <- -1
A3
try(Cholesky(A3, perm = FALSE)) # fails as not positive definite
ch.A3 <- Cholesky(A3) # returns, with a warning and ...
A3.hat <- Reduce(`%*%`, expand2(ch.A3, LDL = FALSE))
norm(A3 - A3.hat, "2") / norm(A3, "2") # 1.781568

## Indeed, 'A3' is not positive semidefinite, but 'A3.hat' _is_
ch.A3.hat <- Cholesky(A3.hat)
A3.hat.hat <- Reduce(`%*%`, expand2(ch.A3.hat, LDL = FALSE))
norm(A3.hat - A3.hat.hat, "2") / norm(A3.hat, "2") # 1.777944e-16

## ---- Sparse -----

## Really just three cases modulo permutation :
##
##           type           factorization  minors of P1 A P1'
## 1 simplicial P1 A P1' = L1 D L1'         nonzero
## 2 simplicial P1 A P1' = L   L '         positive
## 3 supernodal P1 A P2' = L   L '         positive

data(KNex, package = "Matrix")
A4 <- crossprod(KNex[["mm"]])

ch.A4 <-
list(pivoted =
  list(simpl1 = Cholesky(A4, perm = TRUE, super = FALSE, LDL = TRUE),

```

```

        simpl0 = Cholesky(A4, perm = TRUE, super = FALSE, LDL = FALSE),
        super0 = Cholesky(A4, perm = TRUE, super = TRUE
    )),
    unpivoted =
    list(simpl1 = Cholesky(A4, perm = FALSE, super = FALSE, LDL = TRUE),
        simpl0 = Cholesky(A4, perm = FALSE, super = FALSE, LDL = FALSE),
        super0 = Cholesky(A4, perm = FALSE, super = TRUE
    )))
ch.A4

s <- simplify2array
rapply2 <- function(object, f, ...) rapply(object, f, , , how = "list", ...)

s(rapply2(ch.A4, isLDL))
s(m.ch.A4 <- rapply2(ch.A4, expand1, "L")) # giving L = L1 sqrt(D)

## By design, the pivoted and simplicial factorizations
## are more sparse than the unpivoted and supernodal ones ...
s(rapply2(m.ch.A4, object.size))

## Which is nicely visualized by lattice-based methods for 'image'
inm <- c("pivoted", "unpivoted")
jnm <- c("simpl1", "simpl0", "super0")
for(i in 1:2)
  for(j in 1:3)
    print(image(m.ch.A4[[c(i, j)]]), main = paste(inm[i], jnm[j])),
          split = c(j, i, 3L, 2L), more = i * j < 6L)

simpl1 <- ch.A4[[c("pivoted", "simpl1")]]
stopifnot(exprs = {
  length(simpl1@perm) == ncol(A4)
  isPerm(simpl1@perm, 0L)
  is.unsorted(simpl1@perm) # typically not the identity permutation
})

## One can expand with and without D regardless of isLDL(.),
## but "without" requires L = L1 sqrt(D), which is conditional
## on min(diag(D)) >= 0, hence "with" is the default
isLDL(simpl1)
stopifnot(min(diag(simpl1)) >= 0)
str(e.ch.A4 <- expand2(simpl1, LDL = TRUE), max.level = 2L) # default
str(E.ch.A4 <- expand2(simpl1, LDL = FALSE), max.level = 2L)
stopifnot(exprs = {
  all.equal(E.ch.A4[["L" ]], e.ch.A4[["L1" ]] %*% sqrt(e.ch.A4[["D"]]))
  all.equal(E.ch.A4[["L." ]], sqrt(e.ch.A4[["D"]]) %*% e.ch.A4[["L1." ]])
  all.equal(A4, as(Reduce(`%*%`, e.ch.A4), "symmetricMatrix"))
  all.equal(A4, as(Reduce(`%*%`, E.ch.A4), "symmetricMatrix"))
})

## The "same" permutation matrix with "alternate" representation
## [i, perm[i]] {margin=1} <-> [invertPerm(perm)[j], j] {margin=2}
alt <- function(P) {
  P@margin <- 1L + !(P@margin - 1L) # 1 <-> 2
  P@perm <- invertPerm(P@perm)
  P
}

## Expansions are elegant but inefficient (transposes are redundant)
## hence programmers should consider methods for 'expand1' and 'diag'

```

```

stopifnot(exprs = {
  identical(expand1(simpl1, "P1"), alt(e.ch.A4[["P1"]]))
  identical(expand1(simpl1, "L"), E.ch.A4[["L"]])
  identical(Diagonal(x = diag(simpl1)), e.ch.A4[["D"]])
})

## chol(A, pivot = value) is a simple wrapper around
## Cholesky(A, perm = value, LDL = FALSE, super = FALSE),
## returning L' = sqrt(D) L1' _but_ giving no information
## about the permutation P1
selectMethod("chol", "dsCMatrix")
stopifnot(all.equal(chol(A4, pivot = TRUE), E.ch.A4[["L."]]))

## Now a symmetric matrix with positive _and_ negative eigenvalues,
## hence _not_ positive semidefinite
A5 <- new("dsCMatrix",
  Dim = c(7L, 7L),
  p = c(0:1, 3L, 6:7, 10:11, 15L),
  i = c(0L, 0:1, 0:3, 2:5, 3:6),
  x = c(1, 6, 38, 10, 60, 103, -4, 6, -32, -247, -2, -16, -128, -2, -67))
(ev <- eigen(A5, only.values = TRUE)$values)
(t.ev <- table(factor(sign(ev), -1:1))) # the matrix "inertia"

ch.A5 <- Cholesky(A5)
isLDL(ch.A5)
(d.A5 <- diag(ch.A5)) # diag(D) is partly negative

## Sylvester's law of inertia holds here, but not in general
## in finite precision arithmetic
stopifnot(identical(table(factor(sign(d.A5), -1:1)), t.ev))

try(expand1(ch.A5, "L")) # unable to compute L = L1 sqrt(D)
try(expand2(ch.A5, LDL = FALSE)) # ditto
try(chol(A5, pivot = TRUE)) # ditto

## The default expansion is "square root free" and still works here
str(e.ch.A5 <- expand2(ch.A5, LDL = TRUE), max.level = 2L)
stopifnot(all.equal(A5, as(Reduce(`%*%`, e.ch.A5), "symmetricMatrix")))

## Version of the SuiteSparse library, which includes CHOLMOD
Mv <- Matrix.Version()
Mv[["suitesparse"]]

```

coerce-methods-graph

[graphT2graph](#)[graph2Tas](#)(from, "<Class>")

```

graph2T(from, use.weights = )
T2graph(from, need.uniq = !isUniqueT(from), edgemode = NULL)

```

```

from          graph2T()"graph"
              T2graph()"TsparseMatrix"

use.weights   dgTMatrixngTMatrixnsTMatrix1

need.uniq     from

edgemode      NULL"directed""undirected"NULL"undirected"

```

```

graph2T()"TsparseMatrix"
T2graph()"graph"

```

```

igraphgraph_from_adjacency_matrixas_adjacency_matrix

```

```

if(requireNamespace("graph")) {
  n4 <- LETTERS[1:4]; dns <- list(n4,n4)
  show(a1 <- sparseMatrix(i= c(1:4), j=c(2:4,1), x = 2, dimnames=dns))
  show(g1 <- as(a1, "graph")) # directed
  unlist(graph::edgeWeights(g1)) # all '2'

  show(a2 <- sparseMatrix(i= c(1:4,4), j=c(2:4,1:2), x = TRUE, dimnames=dns))
  show(g2 <- as(a2, "graph")) # directed
  # now if you want it undirected:
  show(g3 <- T2graph(as(a2,"TsparseMatrix"), edgemode="undirected"))
  show(m3 <- as(g3,"Matrix"))
  show( graph2T(g3) ) # a "pattern Matrix" (nsTMatrix)

  a. <- sparseMatrix(i=4:1, j=1:4, dimnames=list(n4, n4), repr="T") # no 'x'
  show(a.) # "ngTMatrix"
  show(g. <- as(a., "graph"))
}

```

coerce-methods-SparseM

```

as(., "<class>")

as(from, Class)

```

[SparseM.ontologymatrix.csr](#)

colSums-methods

[sparseMatrix](#)[sparseVector](#)[colSums](#)[numeric](#)

```
colSums(x, na.rm = FALSE, dims = 1L, ...)  
rowSums(x, na.rm = FALSE, dims = 1L, ...)  
colMeans(x, na.rm = FALSE, dims = 1L, ...)  
rowMeans(x, na.rm = FALSE, dims = 1L, ...)
```

```
## S4 method for signature 'CsparseMatrix'  
colSums(x, na.rm = FALSE, dims = 1L,  
        sparseResult = FALSE, ...)  
## S4 method for signature 'CsparseMatrix'  
rowSums(x, na.rm = FALSE, dims = 1L,  
        sparseResult = FALSE, ...)  
## S4 method for signature 'CsparseMatrix'  
colMeans(x, na.rm = FALSE, dims = 1L,  
         sparseResult = FALSE, ...)  
## S4 method for signature 'CsparseMatrix'  
rowMeans(x, na.rm = FALSE, dims = 1L,  
         sparseResult = FALSE, ...)
```

```
x           Matrix  
na.rm       NaN  
dims        Matrix  
...         <->  
sparseResult sparseVectorxsparseMatrix
```

```
sparseResult FALSEsparseVector  
dimnames(x)names(v)vnumericsparseVector
```

colSumssparseVector

```
(M <- bdiag(Diagonal(2), matrix(1:3, 3,4), diag(3:2))) # 7 x 8
colSums(M)
d <- Diagonal(10, c(0,0,10,0,0,2,rep(0,5)))
MM <- kronecker(d, M)
dim(MM) # 70 80
length(MM@x) # 160, but many are '0' ; drop those:
MM <- drop0(MM)
length(MM@x) # 32
cm <- colSums(MM)
(scm <- colSums(MM, sparseResult = TRUE))
stopifnot(is(scm, "sparseVector"),
           identical(cm, as.numeric(scm)))
rowSums(MM, sparseResult = TRUE) # 14 of 70 are not zero
colMeans(MM, sparseResult = TRUE) # 16 of 80 are not zero
## Since we have no 'NA's, these two are equivalent :
stopifnot(identical(rowMeans(MM, sparseResult = TRUE),
                    rowMeans(MM, sparseResult = TRUE, na.rm = TRUE)),
           rowMeans(Diagonal(16)) == 1/16,
           colSums(Diagonal(7)) == 1)

## dimnames(x) --> names( <value> ) :
dimnames(M) <- list(paste0("r", 1:7), paste0("V",1:8))
M
colSums(M)
rowMeans(M)
## Assertions :
stopifnot(exprs = {
  all.equal(colSums(M),
            structure(c(1,1,6,6,6,6,3,2), names = colnames(M)))
  all.equal(rowMeans(M),
            structure(c(1,1,4,8,12,3,2)/8, names = paste0("r", 1:7)))
})
```

condest

```
Anorm(A,"1")onenormest(.)
```

```
condest(A, t = min(n, 5), normA = norm(A, "1"),
        silent = FALSE, quiet = TRUE)
```

```
onenormest(A, t = min(n, 5), A.x, At.x, n,
           silent = FALSE, quiet = silent,
           iter.max = 10, eps = 4 * .Machine$double.eps)
```



```
CsparseMatrixvalidObjecti
<= 0.999375-16validObjectnew()ixnew(...)
sparseMatrix.validateCsparse()
```

```
colSumskronecker
```

```
CsparseMatrixsparseMatrixdgCMatrix
```

```
getClass("CsparseMatrix")
```

```
## The common validity check function (based on C code):
getValidity(getClass("CsparseMatrix"))
```

ddenseMatrix-class

```
dgeMatrix
```

```
"dMatrix""Matrix"
```

```
dgeMatrix
```

```
as(*, "generalMatrix")
```

```
showMethods(class = "ddenseMatrix", where = "package:Matrix")
```

```
MatrixdMatrixdsparseMatrix
```

```
showClass("ddenseMatrix")
```

```
showMethods(class = "ddenseMatrix", where = "package:Matrix")
```

ddiMatrix-class

"ddiMatrix"sparseMatrix

new("ddiMatrix", ...)Diagonal

x $n \times n$ diag

diag "character""U""N""U"

DimDimnames dimnamesMatrix

"diagonalMatrix""dMatrix""sparseMatrix"showClass("ddiMatrix")

signature(x = "ddiMatrix", y = "ddiMatrix")

diagonalMatrixDiagonal

```
(d2 <- Diagonal(x = c(10,1)))  
str(d2)  
## slightly larger in internal size:  
str(as(d2, "sparseMatrix"))
```

```
M <- Matrix(cbind(1,2:4))  
M %*% d2 #> `fast' multiplication
```

```
chol(d2) # trivial  
stopifnot(is(cd2 <- chol(d2), "ddiMatrix"),  
  all.equal(cd2@x, c(sqrt(10),1)))
```

denseLU-class

denseLU $m \times n$ A

$$P_1 A = LU$$

$$A = P_1' LU$$

$$P_1 m \times m L m \times \min(m, n) U \min(m, n) \times n m = n LU$$

DimDimnames MatrixFactorization

x prod(Dim)LUdgetrf

perm min(Dim) P_1 asPerm(perm)

LUMatrixFactorizationLU

```
new("denseLU", ...)lu(x)xdenseMatrixdgeMatrix
```

```
coerce signature(from = "denseLU", to = "dgeMatrix")dgeMatrix $ALU$ 
```

```
determinant signature(from = "denseLU", logarithm = "logical")A
```

```
expand signature(x = "denseLU")expand-methods
```

```
expand1 signature(x = "denseLU")expand1-methods
```

```
expand2 signature(x = "denseLU")expand2-methods
```

```
solve signature(a = "denseLU", b = "missing")solve-methods
```

<https://netlib.org/lapack/double/dgetrf.f>

sparseLU

dgeMatrix

luexpand1expand2

```
showClass("denseLU")
```

```
set.seed(1)
```

```
n <- 3L
```

```
(A <- Matrix(round(rnorm(n * n), 2L), n, n))
```

```
## With dimnames, to see that they are propagated :
```

```
dimnames(A) <- dn <- list(paste0("r", seq_len(n)),  
                          paste0("c", seq_len(n)))
```

```
(lu.A <- lu(A))
```

```
str(e.lu.A <- expand2(lu.A), max.level = 2L)
```

```
## Underlying LAPACK representation
```

```
(m.lu.A <- as(lu.A, "dgeMatrix")) # which is L and U interlaced
```

```
stopifnot(identical(as(m.lu.A, "matrix"), `dim<-`(lu.A@x, lu.A@Dim)))
```

```
ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
```

```
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)
```

```
## A ~ P1' L U in floating point
```

```
stopifnot(exprs = {
```

```
  identical(names(e.lu.A), c("P1.", "L", "U"))
```

```
  identical(e.lu.A[["P1."],
```

```
    new("pMatrix", Dim = c(n, n), Dimnames = c(dn[1L], list(NULL)),
```

```
    margin = 1L, perm = invertPerm(asPerm(lu.A@perm))))
```

```

    identical(e.lu.A[["L"]],
              new("dtrMatrix", Dim = c(n, n), Dimnames = list(NULL, NULL),
                  uplo = "L", diag = "U", x = lu.A@x))
    identical(e.lu.A[["U"]],
              new("dtrMatrix", Dim = c(n, n), Dimnames = c(list(NULL), dn[2L]),
                  uplo = "U", diag = "N", x = lu.A@x))
    ae1(A, with(e.lu.A, P1. %*% L %*% U))
    ae2(A[asPerm(lu.A@perm), ], with(e.lu.A, L %*% U))
  })

## Factorization handled as factorized matrix
b <- rnorm(n)
stopifnot(identical(det(A), det(lu.A)),
           identical(solve(A, b), solve(lu.A, b)))

```

denseMatrix-class

[packedMatrix](#)[unpackedMatrix](#)[ddenseMatrix](#)[ldenseMatrix](#)[ndenseMatrix](#)

$\text{denseMatrix}_2 + 3 = 5$

"Matrix"

"Matrix" "Dim" "Dimnames"

[showMethods](#)(class = "denseMatrix", where = "package:Matrix")

"["[-methods

[colSums](#)[kronecker](#)

[Matrix](#)[ddenseMatrix](#)[sparseMatrix](#)

[showClass](#)("denseMatrix")

dgCMatrix-class

[dgCMatrix](#)[dgCMatrix](#)

[new](#)("dgCMatrix", ...) [as](#)(*, "CsparseMatrix") [Matrix](#)(*, sparse = TRUE)
[sparseMatrix](#)()

```
x "numeric"
"CsparseMatrix"
```

```
signature(from = "matrix", to = "dgCMatrix")
signature(x = "dgCMatrix")x
signature(x = "dgCMatrix")x
signature(x = "dgCMatrix")xlevelplot
signature(a = "dgCMatrix", b = "...")solve-methodsparse
signature(x = "dgCMatrix")dgCMatrix
```

[dsCMatrix](#)[dtCMatrix](#)[lu](#)

```
(m <- Matrix(c(0,0,2:0), 3,5))
str(m)
m[,1]
```

dgeMatrix-class

dgeMatrix

```
new("dgeMatrix", ...)MatrixMatrix(..)
```

```
x "numeric"
Dim "integer"
Dimnames Matrix
factors "list"
```

[Arith](#)

```
signature(e1 = "dgeMatrix", e2 = "dgeMatrix")
signature(e1 = "dgeMatrix", e2 = "numeric")
signature(e1 = "numeric", e2 = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix", digits = "numeric")
```

```

%%crossprod()tcrossprod()solve
signature(x = "dgeMatrix", vectors = "logical")
signature(x = "dgeMatrix", vectors = "missing")
signature(x = "dgeMatrix")chol
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix", only.values= "logical")
signature(x = "dgeMatrix", only.values= "missing")
signature(x = "dgeMatrix", type = "character")
signature(x = "dgeMatrix", type = "missing")
signature(x = "dgeMatrix", norm = "character")norm = "missing"rcond()
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")
signature(x = "dgeMatrix")

```

MatrixdtrMatrixdsyMatrix

dgRMatrix-class

```

dgRMatrix
dgCMatrix

new("dgRMatrix", ...)

j "integer"
p "integer"
x "numeric"
Dim "integer"

signature(x = "dgRMatrix")x
signature(x = "dgRMatrix")x
signature(x = "dgRMatrix")xlevelplot

```

RsparseMatrixdgCMatrix

dgTMatrix-class

"dgTMatrix"**dgCMatrix**

new("dgTMatrix", ...) **spMatrix()****sparseMatrix**(*, repr = "T")

i **integer**0:(nrow(.)-1)
j **integer**i0:(ncol(.)-1)
x **numeric**(i,j)ix
Dim "integer"

signature(e1 = "dgTMatrix", e2 = "dgTMatrix")
signature(x = "dgTMatrix")**xlevelplot**
signature(x = "dgTMatrix")x

dgCMatrix

new(.)**spMatrix**"dgTMatrix""**TsparseMatrix**" $x_k(i_k, j_k)$
"**TsparseMatrix**"**asUniqueT**()

dgCMatrix**dsparseMatrix****TsparseMatrix****asUniqueT**

```
m <- Matrix(0+1:28, nrow = 4)
m[-3,c(2,4:5,7)] <- m[ 3, 1:4] <- m[1:3, 6] <- 0
(mT <- as(m, "TsparseMatrix"))
str(mT)
mT[1,]
mT[4, drop = FALSE]
stopifnot(identical(mT[lower.tri(mT)],
                    m [lower.tri(m) ]))
mT[lower.tri(mT,diag=TRUE)] <- 0
mT

## Triplet representation with repeated (i,j) entries
## *adds* the corresponding x's:
T2 <- new("dgTMatrix",
          i = as.integer(c(1,1,0,3,3)),
          j = as.integer(c(2,2,4,0,0)), x=10*1:5, Dim=4:5)
str(T2) # contains (i,j,x) slots exactly as above, but
T2 ## has only three non-zero entries, as for repeated (i,j)'s,
      ## the corresponding x's are "implicitly" added
stopifnot(nnzero(T2) == 3)
```

Diagonal

[MatrixdiagonalMatrixCsparseMatrix](#)

`Diagonal(n, x = NULL, names = FALSE)`

```
.sparseDiagonal(n, x = NULL, uplo = "U", shape = "t", unitri = TRUE, kind, cols)
  .trDiagonal(n, x = NULL, uplo = "U", unitri = TRUE, kind)
  .symDiagonal(n, x = NULL, uplo = "U", kind)
```

<code>n</code>	<code>length(x)</code>
<code>x</code>	<code>NULL.sparseDiagonal()NULLxkind = "n"</code>
<code>names</code>	logicalTRUEFALSEcharacterlengthnnames(x)NULL
<code>uplo</code>	<code>c("U","L")uplo</code>
<code>shape</code>	<code>c("t","s","g")</code> triangularMatrixsymmetricMatrixgeneralMatrix
<code>unitri</code>	<code>diag"U""N"</code>
<code>kind</code>	<code>c("d","l","n")</code> dsparseMatrixlsparseMatrixnsparseMatrix "n"xNULL typeof(x)
<code>cols</code>	<code>0:(n-1)D[, cols+1]DD = Diagonal(n, x)shape</code>

`Diagonal()`[diagonalMatrix](#)

```
.sparseDiagonal()CsparseMatrixDiagonal(n, x)colsDiagonal(n, x)[, cols+1]shape
kind
```

```
.trDiagonal().symDiagonal().sparseDiagonal(shape = "t").sparseDiagonal(shape =
"s")
```

```
.sparseDiagonal()CsparseMatrix
```

[diag](#)

[bandSparseband\(A\)A](#)

[MatrixdiagonalMatrix](#)


```

Diagonal(3)
Diagonal(x = 10^(3:1))
Diagonal(x = (1:4) >= 2)#-> "ldiMatrix"

## Use Diagonal() + kronecker() for "repeated-block" matrices:
M1 <- Matrix(0+0:5, 2,3)
(M <- kronecker(Diagonal(3), M1))

(S <- crossprod(Matrix(rbinom(60, size=1, prob=0.1), 10,6)))
(SI <- S + 10*.symDiagonal(6)) # sparse symmetric still
stopifnot(is(SI, "dsCMatrix"))
(I4 <- .sparseDiagonal(4, shape="t"))# now (2012-10) unittriangular
stopifnot(I4@diag == "U", all(I4 == diag(4)))

```

diagonalMatrix-class

```

diag character"U""N""U"
Dim
Dimnames dimnameslistMatrixlist(NULL,NULL)

```

["sparseMatrix"](#)

```

signature(from = "matrix", to = "diagonalMatrix")
signature(from = "Matrix", to = "diagonalMatrix")
signature(from = "diagonalMatrix", to = "generalMatrix")
signature(from = "diagonalMatrix", to = "triangularMatrix")
signature(from = "diagonalMatrix", to = "nMatrix")
signature(from = "diagonalMatrix", to = "matrix")
signature(from = "diagonalMatrix", to = "sparseVector")
signature(x = "diagonalMatrix")

signature(a = "diagonalMatrix", b, ...) solve-methods
signature(x = "nMatrix") which(x, arr.ind)
signature(x = "diagonalMatrix") "diagonalMatrix"cumsum\(\) matrix
signature(e1 = "ddiMatrix", e2="denseMatrix") Ops
signature(e1 = "ddiMatrix", e2="denseMatrix") ddiMatrixe2NA  $0/x = 00/0 \mapsto$ 
(object = "diagonalMatrix") "diagSummary" object@xprint

```

```
Diagonal()isDiagonalddiMatrixldiMatrix"diagonalMatrix"
```

```
I5 <- Diagonal(5)
D5 <- Diagonal(x = 10*(1:5))
## trivial (but explicitly defined) methods:
stopifnot(identical(crossprod(I5), I5),
           identical(tcrossprod(I5), I5),
           identical(crossprod(I5, D5), D5),
           identical(tcrossprod(D5, I5), D5),
           identical(solve(D5), solve(D5, I5)),
           all.equal(D5, solve(solve(D5)), tolerance = 1e-12)
)
solve(D5)# efficient as is diagonal

# an unusual way to construct a band matrix:
rbind2(cbind2(I5, D5),
       cbind2(D5, I5))
```

diagU2N

```
xclasstriangularMatrixdiagU2N(x)diagN2U(x)diagN2U(x)diag(x)
.diagU2N(x).diagN2U(x)xtriangularMatrixdiag"U""N"
```

```
diagU2N(x, cl = getClassDef(class(x)), checkDense = FALSE)
diagN2U(x, cl = getClassDef(class(x)), checkDense = FALSE)

.diagU2N(x, cl = getClassDef(class(x)), checkDense = FALSE)
.diagN2U(x, cl = getClassDef(class(x)), checkDense = FALSE)
```

```
x          triangularMatrix
cl          x
checkDense denseMatrixx
```

```
diag"U"
```

```
classdiagdiagU2NxdiaN2U(x)1
```

```
"triangularMatrix""dtCMatrix"
```

```

(T <- Diagonal(7) + triu(Matrix(rpois(49, 1/4), 7, 7), k = 1))
(uT <- diagN2U(T)) # "unitriangular"
(t.u <- diagN2U(10*T))# changes the diagonal!
stopifnot(all(T == uT), diag(t.u) == 1,
           identical(T, diagU2N(uT)))
T[upper.tri(T)] <- 5 # still "dtC"
T <- diagN2U(as(T,"triangularMatrix"))
dT <- as(T, "denseMatrix") # (unitriangular)
dT.n <- diagU2N(dT, checkDense = TRUE)
sT.n <- diagU2N(dT)
stopifnot(is(dT.n, "denseMatrix"), is(sT.n, "sparseMatrix"),
          dT@diag == "U", dT.n@diag == "N", sT.n@diag == "N",
          all(dT == dT.n), all(dT == sT.n))

```

dimScale

```

dimScalerowScalecolScaleD1 %%% x %%% D2D %%% xx %%% DD1D2Dd1d2ddimnames(x)

```

```

dimScale(x, d1 = sqrt(1/diag(x, names = FALSE)), d2 = d1)
rowScale(x, d)
colScale(x, d)

```

```

x          Matrix
d1d2d      x

```

```

dimScale(x)d1d2cov2cor(x)cov2cordimScale

```

```

xdMatrix

```

```

triangularMatrixxdimScale(x, d1, d2)d1d2symmetricMatrixx

```

```

cov2cor

```

```

n <- 6L
(x <- forceSymmetric(matrix(1, n, n)))
dimnames(x) <- rep.int(list(letters[seq_len(n)]), 2L)

d <- seq_len(n)
(D <- Diagonal(x = d))

(scx <- dimScale(x, d)) # symmetry and 'dimnames' kept
(mmx <- D %*% x %*% D) # symmetry and 'dimnames' lost
stopifnot(identical(unname(as(scx, "generalMatrix")), mmx))

rowScale(x, d)
colScale(x, d)

```

dMatrix-class

dMatrix1Matrix

Dim "integer"
 Dimnames [character](#)Dim

[Arith](#)

```

signature(e1 = "dMatrix", e2 = "dMatrix")
signature(e1 = "dMatrix", e2 = "numeric")
signature(e1 = "numeric", e2 = "dMatrix")
signature(x = "dMatrix")
signature(x = "dMatrix", digits = "numeric")round\(\)signif\(\)
signature(e1 = "numeric", e2 = "dMatrix")
signature(e1 = "dMatrix", e2 = "numeric")
signature(e1 = "dMatrix", e2 = "dMatrix")
signature(x = "dMatrix")"Summary"max\(\)min\(\)range\(\)prod\(\)sum\(\)any\(\)all\(\)

```

```
signature(x = "dMatrix")expm
```

```
signature(x = "lsparseMatrix")"1Matrix"which(x, arr.ind)TRUExarr.induseNames
dimnamesbase::which
```

```
nMatrixNAlogical
dgeMatrixdgCMatrixMatrix
drop0(x, tol=1e-10)zapsmall(x, digits=10)
```

```
showClass("dMatrix")

set.seed(101)
round(Matrix(rnorm(28), 4,7), 2)
M <- Matrix(rlnorm(56, sd=10), 4,14)
(M. <- zapsmall(M))
table(as.logical(M. == 0))
```

dmperm

$n \times m \times n$

```
dmperm(x, nAns = 6L, seed = 0L)
```

x	"dgCMatrix""dtCMatrix"
nAns	lengthlist
seed	seed = 0seed = -1k:1seed = 1

list

p	pnrow(x)
q	qncol(x)
r	nb+1
s	nb+1
rr5	
cc5	

Schur"pMatrix"

```
set.seed(17)
(S9 <- rsparsematrix(9, 9, nnz = 10, symmetric=TRUE)) # dsCMatrix
str( dm9 <- dmperm(S9) )
(S9p <- with(dm9, S9[p, q]))
## looks good, but *not* quite upper triangular; these, too:
str( dm9.0 <- dmperm(S9, seed=-1)) # non-random too.
str( dm9_1 <- dmperm(S9, seed= 1)) # a random one
## The last two permutations differ, but have the same effect!
(S9p0 <- with(dm9.0, S9[p, q])) # .. hmm ..
stopifnot(all.equal(S9p0, S9p))# same as as default, but different from the random one
```

```
set.seed(11)
(M <- triu(rsparsematrix(9,11, 1/4)))
dM <- dmperm(M); with(dM, M[p, q])
(Mp <- M[sample.int(nrow(M)), sample.int(ncol(M))])
dMp <- dmperm(Mp); with(dMp, Mp[p, q])
```

```
set.seed(7)
(n7 <- rsparsematrix(5, 12, nnz = 10, rand.x = NULL))
str( dm.7 <- dmperm(n7) )
stopifnot(exprs = {
  lengths(dm.7[1:2]) == dim(n7)
  identical(dm.7, dmperm(as(n7, "dMatrix")))
  identical(dm.7[1:4], dmperm(n7, nAns=4))
  identical(dm.7[1:2], dmperm(n7, nAns=2))
})
```

dpoMatrix-class

```
"dpoMatrix"
"dppMatrix"
"corMatrix""copMatrix""dpoMatrix""dppMatrix"sd
```

```
new("dpoMatrix", ...)crossprod"dgeMatrix"
```

```
uplo "character"
x "numeric"
Dim "integer"
Dimnames "Matrix"
factors "list"
sd "corMatrix""copMatrix"numeric $\sqrt{\text{var}(.)}$ 
```

```
"dsyMatrix"
"dgeMatrix""symmetricMatrix""dsyMatrix"
```

```
signature(x = "dpoMatrix")xchol
signature(x = "dpoMatrix")determinantxchol(x)
signature(x = "dpoMatrix", norm = "character")xnorm"0""I"
signature(a = "dpoMatrix", b = "...")
signature(a = "dppMatrix", b = "...")solve-methods
signature(e1 = "dpoMatrix", e2 = "numeric")length(.) == 1logical
```

```
getValidity(getClass("dpoMatrix"))
diag(BunchKaufman(.))!inherits(tryCatch(chol(.), error=identity), "error")
Cholesky()chol()selectMethod("coerce", c(from="dsyMatrix", to="dpoMatrix"))
```

```
dsyMatrixdgeMatrixMatrixrcondcholsolvecrossprod
```

```
h6 <- Hilbert(6)
rcond(h6)
str(h6)
h6 * 27720 # is ``integer``
solve(h6)
str(hp6 <- pack(h6))

### Note that as(*, "corMatrix") *scales* the matrix
(ch6 <- as(h6, "corMatrix"))
stopifnot(all.equal(as(h6 * 27720, "dsyMatrix"), round(27720 * h6),
  tolerance = 1e-14),
  all.equal(ch6@sd^(-2), 2*(1:6)-1,
  tolerance = 1e-12))
chch <- Cholesky(ch6, perm = FALSE)
stopifnot(identical(chch, ch6@factors$Cholesky),
  all(abs(crossprod(as(chch, "dtrMatrix")) - ch6) < 1e-10))
```

drop0

```
drop0(x, tol = 0, is.Csparse = NA, give.Csparse = TRUE)
```

```

x          MatrixsparseMatrixdenseMatrixCsparseMatrixxas(x,
                  "CsparseMatrix")tol = 0
tol        xtol
is.Csparse give.CsparseTRUExCsparseMatrix
give.Csparse CsparseMatrixFALSExRsparseMatrixTsparseMatrixindMatrixxTRUE

```

```
sparseMatrixx
```

```
drop0zapsmall
```

```
sparseMatrixsparseMatrixnnzero
```

```

(m <- sparseMatrix(i = 1:8, j = 2:9, x = c(0:2, 3:-1),
                  dims = c(10L, 20L)))
drop0(m)

## A larger example:
t5 <- new("dtCMatrix", Dim = c(5L, 5L), uplo = "L",
          x = c(10, 1, 3, 10, 1, 10, 1, 10, 10),
          i = c(0L,2L,4L, 1L, 3L,2L,4L, 3L, 4L),
          p = c(0L, 3L, 5L, 7:9))
TT <- kronecker(t5, kronecker(kronecker(t5, t5), t5))
IT <- solve(TT)
I. <- TT %*% IT ; nnzero(I.) # 697 ( == 625 + 72 )
I.0 <- drop0(zapsmall(I.))
## which actually can be more efficiently achieved by
I.. <- drop0(I., tol = 1e-15)
stopifnot(all(I.0 == Diagonal(625)), nnzero(I..) == 625)

```

dsCMatrix-class

dsCMatrix

dsTMatrix

```

new("dsCMatrix", ...)new("dsTMatrix", ...)as(*, "symmetricMatrix")dsCMatrix
Matrix(.)
sparseMatrix(*, symmetric=TRUE)

```



```

uplo "U""L"
i "integer"
p "dsCMatrix"integerCsparseMatrix
j "dsTMatrix""integer"i
x "numeric"
factors "list"
Dim "integer"

```

[symmetricMatrix](#)[dsparseMatrix](#)[dsCMatrix](#)[CsparseMatrix](#)[dsTMatrix](#)[TsparseMatrix](#)

```

signature(a = "dsCMatrix", b = "...")x <- solve(a,b) $Ax = bx$ solve-methods
signature(x = "dsCMatrix", pivot = "logical")xchol
signature(A = "dsCMatrix",...)Cholesky
signature(x = "dsCMatrix", logarithm = "missing")x
signature(x = "dsCMatrix", logarithm = "logical")xlogarithm
signature(x = "dsCMatrix")uplo
signature(x = "dsTMatrix")uplo"U""L""dsCMatrix"

```

[dgCMatrix](#)[dgTMatrix](#)[dgeMatrix](#)

```

mm <- Matrix(toeplitz(c(10, 0, 1, 0, 3)), sparse = TRUE)
mm # automatically dsCMatrix
str(mm)
mT <- as(as(mm, "generalMatrix"), "TsparseMatrix")

## Either
(symM <- as(mT, "symmetricMatrix")) # dsT
(symC <- as(symM, "CsparseMatrix")) # dsC
## or
sT <- Matrix(mT, sparse=TRUE, forceCheck=TRUE) # dsT

sym2 <- as(symC, "TsparseMatrix")
## --> the same as 'symM', a "dsTMatrix"

```

dsparseMatrix-class

```
"dsparseMatrix"
```

```
Dim "Matrix"  
Dimnames "Matrix"  
x numeric
```

```
"dMatrix""sparseMatrix"  
"Matrix"
```

```
showClass("dsparseMatrix")
```

```
showClass("dsparseMatrix")
```

dsRMatrix-class

```
dsRMatrix
```

```
"..RMatrix"  
new("dsRMatrix", ...)
```

```
uplo "U""L"  
j "integer"nnzero  
p "integer"  
factors "list"  
x "numeric"  
Dim "integer"  
Dimnames Matrix
```

```
dsparseMatrixsymmetricMatrixRsparseMatrix  
"dMatrix""dsparseMatrix""sparseMatrix""dsparseMatrix""RsparseMatrix"
```

```
signature(x = "dsRMatrix", uplo = "missing")x
signature(x = "dsRMatrix", uplo = "character")uplo == x@uploxt(x)
```

[dgCMatrix](#)[dgTMatrix](#)[dgeMatrix](#)

```
(m0 <- new("dsRMatrix"))
m2 <- new("dsRMatrix", Dim = c(2L,2L),
          x = c(3,1), j = c(1L,1L), p = 0:2)
m2
stopifnot(colSums(as(m2, "TsparseMatrix")) == 3:4)
str(m2)
(ds2 <- forceSymmetric(diag(2))) # dsy*
dR <- as(ds2, "RsparseMatrix")
dR # dsRMatrix
```

dsyMatrix-class

```
"dsyMatrix"
"dspMatrix"pack\(\)
```

```
new("dsyMatrix", ...)new("dspMatrix", ...)
```

```
uplo "character"
x "numeric"
DimDimnames "integer"NULLMatrix
factors "list"
```

```
"dsyMatrix""dgeMatrix"
"dspMatrix""ddenseMatrix"
"symmetricMatrix""Matrix"showClass("dsyMatrix")
```

```
signature(x = "dspMatrix", type = "character")x = "dsyMatrix"type = "missing"
norm
signature(x = "dspMatrix", type = "character")x = "dsyMatrix"type = "missing"
rcond\(\)
signature(a = "dspMatrix", b = "....")
signature(a = "dsyMatrix", b = "....")x <- solve(a,b) $Ax = bx$ solve-methods
signature(x = "dsyMatrix")"U""L"
```

[dpoMatrix](#)[dppMatrix](#)[corMatrix](#)
[dgeMatrix](#)[Matrix](#)[solvenormrcondt](#)

```
## Only upper triangular part matters (when uplo == "U" as per default)
(sy2 <- new("dsyMatrix", Dim = as.integer(c(2,2)), x = c(14, NA,32,77)))
str(t(sy2)) # uplo = "L", and the lower tri. (i.e. NA is replaced).

chol(sy2) #-> "Cholesky" matrix
(sp2 <- pack(sy2)) # a "dspMatrix"

## Coercing to dpoMatrix gives invalid object:
sy3 <- new("dsyMatrix", Dim = as.integer(c(2,2)), x = c(14, -1, 2, -7))
try(as(sy3, "dpoMatrix")) # -> error: not positive definite

## 4x4 example
m <- matrix(0,4,4); m[upper.tri(m)] <- 1:6
(sym <- m+t(m)+diag(11:14, 4))
(S1 <- pack(sym))
(S2 <- t(S1))
stopifnot(all(S1 == S2)) # equal "seen as matrix", but differ internally :
str(S1)
S2@x
```

dtCMatrix-class

"dtCMatrix"

"dtTMatrix"

`new("dtCMatrix", ...)``new("dtTMatrix", ...)`[Matrix\(\)](#)`as(x, "triangularMatrix")`

uplo "character"

diag "character""U""N"[triangularMatrix](#)

p "dtCMatrix"[integerCsparseMatrix](#)

i "integer"

j "integer"dtTMatrix

x "numeric"

DimDimnames "integer"NULL[Matrix](#)

"dgCMatrix""triangularMatrix""dMatrix""sparseMatrix""dgCMatrix"

```
signature(a = "dtCMatrix", b = "...")solve-methods
signature(x = "dtCMatrix")x
signature(x = "dtTMatrix")x
```

[dgCMatrix](#)[dgTMatrix](#)[dgeMatrix](#)[dtrMatrix](#)

```
showClass("dtCMatrix")
showClass("dtTMatrix")
t1 <- new("dtTMatrix", x= c(3,7), i= 0:1, j=3:2, Dim= as.integer(c(4,4)))
t1
## from 0-diagonal to unit-diagonal {low-level step}:
tu <- t1 ; tu@diag <- "U"
tu
(cu <- as(tu, "CsparseMatrix"))
str(cu)# only two entries in @i and @x
stopifnot(cu@i == 1:0,
           all(2 * symmpart(cu) == Diagonal(4) + forceSymmetric(cu)))

t1[1,2:3] <- -1:-2
diag(t1) <- 10*c(1:2,3:2)
t1 # still triangular
(it1 <- solve(t1))
t1. <- solve(it1)
all(abs(t1 - t1.) < 10 * .Machine$double.eps)

## 2nd example
U5 <- new("dtCMatrix", i= c(1L, 0:3), p=c(0L,0L,0:2, 5L), Dim = c(5L, 5L),
          x = rep(1, 5), diag = "U")
U5
(iu <- solve(U5)) # contains one '0'
validObject(iu2 <- solve(U5, Diagonal(5)))# failed in earlier versions

I5 <- iu %*% U5 # should equal the identity matrix
i5 <- iu2 %*% U5
m53 <- matrix(1:15, 5,3, dimnames=list(NULL,letters[1:3]))
asDiag <- function(M) as(drop0(M), "diagonalMatrix")
stopifnot(
  all.equal(Diagonal(5), asDiag(I5), tolerance=1e-14) ,
  all.equal(Diagonal(5), asDiag(i5), tolerance=1e-14) ,
  identical(list(NULL, dimnames(m53)[[2]]), dimnames(solve(U5, m53)))
)
```

`dtpMatrix-class`

"dtpMatrix""dtrMatrix"

`new("dtpMatrix", ...)`

```

uplo "character"
diag "character""U""N"triangularMatrix
x "numeric" $d \times d$ length(x) $d(d+1)/2$ diag == "U"
DimDimnames "integer"NULLMatrix

"ddenseMatrix""triangularMatrix""dMatrix""ddenseMatrix"

signature(x = "dtpMatrix", y = "dgeMatrix")showMethods("%*%", class =
  "dtpMatrix")
signature(x = "dtpMatrix", logarithm = "logical")determinant(x)prod(diag(x))
signature(x = "dtpMatrix")
signature(x = "dtpMatrix", type = "character")
signature(x = "dtpMatrix", norm = "character")
signature(a = "dtpMatrix", b = "...")solve-methods
signature(x = "dtpMatrix")t(x)"dtpMatrix"x

```

[dtrMatrix](#)

```

showClass("dtrMatrix")

example("dtrMatrix-class", echo=FALSE)
(p1 <- pack(T2))
str(p1)
(pp <- pack(T))
ip1 <- solve(p1)
stopifnot(length(p1@x) == 3, length(pp@x) == 3,
  p1 @ uplo == T2 @ uplo, pp @ uplo == T @ uplo,
  identical(t(pp), p1), identical(t(p1), pp),
  all((l.d <- p1 - T2) == 0), is(l.d, "dtpMatrix"),
  all((u.d <- pp - T ) == 0), is(u.d, "dtpMatrix"),
  l.d@uplo == T2@uplo, u.d@uplo == T@uplo,
  identical(t(ip1), solve(pp)), is(ip1, "dtpMatrix"),
  all.equal(as(solve(p1,p1), "diagonalMatrix"), Diagonal(2)))

```

dtRMatrix-class

dtRMatrix

```
new("dtRMatrix", ...)
```

```

uplo "character"
diag "character""U""N"triangularMatrix
j "integer"nnzero\(.\)
p "integer"dsRMatrix
x "numeric"
Dim "integer"
Dimnames NULLMatrix

"dgRMatrix""dsparseMatrix""dgRMatrix""dMatrix""dgRMatrix""sparseMatrix"
"dgRMatrix""Matrix""dgRMatrix"

```

[dgCMatrix](#)[dgTMatrix](#)[dgeMatrix](#)

```

(m0 <- new("dtRMatrix"))
(m2 <- new("dtRMatrix", Dim = c(2L,2L),
          x = c(5, 1:2), p = c(0L,2:3), j= c(0:1,1L)))
str(m2)
(m3 <- as(Diagonal(2), "RsparseMatrix"))# --> dtRMatrix

```

dtrMatrix-class

"dtrMatrix""dtpMatrix"[pack\(\)](#)

new("dtrMatrix", ...)

```

uplo "character"
diag "character""U""N"triangularMatrix
x "numeric"
Dim "integer"

```

"ddenseMatrix""triangularMatrix""Matrix""ddenseMatrix"

[?crossprod-methods](#)

```
signature(x = "dtrMatrix", type = "character")
signature(x = "dtrMatrix", norm = "character")
signature(a = "dtrMatrix", b = "...")backsolvesolve-methods
Ops
```

[ddenseMatrixdtpMatrixtriangularMatrix](#)

```
(m <- rbind(2:3, 0:-1))
(M <- as(m, "generalMatrix"))

(T <- as(M, "triangularMatrix")) # formally upper triangular
(T2 <- as(t(M), "triangularMatrix"))
stopifnot(T@uplo == "U", T2@uplo == "L", identical(T2, t(T)))

m <- matrix(0,4,4); m[upper.tri(m)] <- 1:6
(t1 <- Matrix(m+diag(,4)))
str(t1p <- pack(t1))
(t1pu <- diagN2U(t1p))
stopifnot(exprs = {
  inherits(t1, "dtrMatrix"); validObject(t1)
  inherits(t1p, "dtpMatrix"); validObject(t1p)
  inherits(t1pu, "dtCMatrix"); validObject(t1pu)
  t1pu@x == 1:6
  all(t1pu == t1p)
  identical((t1pu - t1)@x, numeric())# sparse all-0
})
```

[expand-methods](#)

[expand1expand2](#)

```
expand1(x, which, ...)
expand2(x, ...)
```

```
expand (x, ...)
```

```
x          MatrixFactorization
which
...
```



```
expand< 1.6-0expand1expand2expand2dimnames
```

```
xy
```

```
all.equal(as(x, "matrix"), as(Reduce(`%*%`, expand2(y)), "matrix"))
```

```
TRUE
```

```
expand1Matrixwhich
```

```
expand2list(L=, U=)Dimnamesx
```

```
expand1which
```

```

class(x)  which
  Schur   c("Q", "T", "Q.")
  denseLU c("P1", "P1.", "L", "U")
  sparseLU c("P1", "P1.", "P2", "P2.", "L", "U")
  sparseQR c("P1", "P1.", "P2", "P2.", "Q", "Q1", "R", "R1")
BunchKaufmanpBunchKaufman c("U", "DU", "U.", "L", "DL", "L.")
  CholeskypCholesky c("P1", "P1.", "L1", "D", "L1.", "L", "L.")
  CHMsimplCHMsimpl c("P1", "P1.", "L1", "D", "L1.", "L", "L.")

```

```
expand2expandexpand1
```

```
expand2 signature(x = "CHMsimpl")  $A = P_1' L_1 D L_1' P_1 = P_1' L L' P_1$  list(P1., L1, D, L1., P1) list(P1., L, L., P1) LDLP1P1.pMatrixL1L1.LL.dtCMatrixDddiMatrix
```

```
expand2 signature(x = "CHMSuper") CHMsimpldgCMatrix
```

```
expand2 signature(x = "p?Cholesky")  $A = L_1 D L_1' = L L'$  list(L1, D, L1.) list(L, L.) LDLL1L1.LL.dtrMatrixdtpMatrixDddiMatrix
```

```
expand2 signature(x = "p?BunchKaufman")  $A = U D_U U' = L D_L L' U = \prod_{k=1}^{b_U} P_k U_k L = \prod_{k=1}^{b_L} P_k L_k$  list(U, DU, U.) list(L, DL, L.) x@uplocompleteTRUE2bU + 12bL + 1Pk pMatrixUkLkdtCMatrixDUDLdsCMatrix
```

```
expand2 signature(x = "Schur")  $A = Q T Q'$  list(Q, T, Q.) QQ.x@Qt(x@Q) DimnamesTx@T
```

```
expand2 signature(x = "sparseLU")  $A = P_1' L U P_2'$  list(P1., L, U, P2.) P1.P2.pMatrixLUdtCMatrix
```

```
expand2 signature(x = "denseLU")  $A = P_1' L U$  list(P1., L, U) P1.pMatrixLUdtrMatrixdgeMatrix
```

```
expand2 signature(x = "sparseQR")  $A = P_1' Q R P_2' = P_1' Q_1 R_1 P_2'$  list(P1., Q, R, P2.) list(P1., Q1, R1, P2.) completeP1.P2.pMatrixQQ1dgeMatrixRdgCMatrixR1dtCMatrix
```

```
expand signature(x = "CHMfactor") expand2list(P, L) expand(x)[["P"]]
expand2(x)[["P1"]] P1marginpermexpand(x)x@Dimnames
```

```
expand signature(x = "sparseLU") expand2list(P, L, U, Q) expand(x)[["Q"]]
expand2(x)[["P2."]] P2marginpermexpand(x)[["P"]] P1P1' expand2(x)[["P1."]]
permexpand(x)[["L"]] expand2(x)[["L"]] Ldiag"N""U" expand(x)[["L"]]
expand(x)[["U"]] x@DimnamesDimnames
```

```
expand signature(x = "denseLU") expand2list(L, U, P) expand(x)[["P"]]
expand2(x)[["P1."]] Dimnamesexpand(x)x@Dimnames
```

MatrixFactorization

CholeskyBunchKaufmanSchurluqr

```
showMethods("expand1", inherited = FALSE)
showMethods("expand2", inherited = FALSE)
set.seed(0)

(A <- Matrix(rnorm(9L, 0, 10), 3L, 3L))
(lu.A <- lu(A))
(e.lu.A <- expand2(lu.A))
stopifnot(exprs = {
  is.list(e.lu.A)
  identical(names(e.lu.A), c("P1.", "L", "U"))
  all(sapply(e.lu.A, is, "Matrix"))
  all.equal(as(A, "matrix"), as(Reduce("%*%", e.lu.A), "matrix"))
})

## 'expand1' and 'expand2' give equivalent results modulo
## dimnames and representation of permutation matrices;
## see also function 'alt' in example("Cholesky-methods")
(a1 <- sapply(names(e.lu.A), expand1, x = lu.A, simplify = FALSE))
all.equal(a1, e.lu.A)

## see help("denseLU-class") and others for more examples
```

expm-methods

expm(x)

x [dMatrix](#)

$\text{expm}(A) = I + A + A^2/2! + A^3/3! + \dots$ [dgeMatrix](#)

x

<A.S.Hodel@Eng.Auburn.EDU>

https://en.wikipedia.org/wiki/Matrix_exponential

<https://mathworld.wolfram.com/MatrixExponential.html>

`expm()``logm()``sqrtn()`

[Schur](#)

```
(m1 <- Matrix(c(1,0,1,1), ncol = 2))
(e1 <- expm(m1)) ; e <- exp(1)
stopifnot(all.equal(e1@x, c(e,0,e,e), tolerance = 1e-15))
(m2 <- Matrix(c(-49, -64, 24, 31), ncol = 2))
(e2 <- expm(m2))
(m3 <- Matrix(cbind(0,rbind(6*diag(3),0))))# sparse!
(e3 <- expm(m3)) # upper triangular
```

`externalFormats`

[sparseMatrix](#)

```
readHB(file)
readMM(file)
writeMM(obj, file, ...)
```

```
obj
file          writeMMreadHBreadMM".rua""".rsa""".mtx"
              readHBreadMM
...
```

```
readHBreadMM"Matrix"writeMMNULLobjfile
```

```
writeHBwriteMM
dimnameswriteMM()
Ssummary(S)data.frameijxsummarysparseMatrix-class
```

<https://math.nist.gov/MatrixMarket/>

<https://sparse.tamu.edu/>

```

str(pores <- readMM(system.file("external/pores_1.mtx", package = "Matrix")))
str(utm <- readHB(system.file("external/utm300.rua" , package = "Matrix")))
str(lundA <- readMM(system.file("external/lund_a.mtx" , package = "Matrix")))
str(lundA <- readHB(system.file("external/lund_a.rsa" , package = "Matrix")))
## https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/counterx/counterx.htm
str(jgl <- readMM(system.file("external/jgl009.mtx" , package = "Matrix")))

## NOTE: The following examples take quite some time
## ---- even on a fast internet connection:
if(FALSE) {
  ## The URL has been corrected, but we need an untar step:
  u. <- url("https://www.cise.ufl.edu/research/sparse/RB/Boeing/msc00726.tar.gz")
  str(sm <- readHB(gzcon(u.)))
}

data(KNex, package = "Matrix")
## Store as MatrixMarket (".mtx") file, here inside temporary dir./folder:
(MMfile <- file.path(tempdir(), "mmMM.mtx"))
writeMM(KNex$mm, file=MMfile)
file.info(MMfile)[,c("size", "ctime")] # (some confirmation of the file's)

## very simple export - in triplet format - to text file:
data(CAex, package = "Matrix")
s.CA <- summary(CAex)
s.CA # shows (i, j, x) [columns of a data frame]
message("writing to ", outf <- tempfile())
write.table(s.CA, file = outf, row.names=FALSE)
## and read it back -- showing off sparseMatrix():
str(dd <- read.table(outf, header=TRUE))
## has columns (i, j, x) -> we can use via do.call() as arguments to sparseMatrix():
mm <- do.call(sparseMatrix, dd)
stopifnot(all.equal(mm, CAex, tolerance=1e-15))

```

facmul-methods

facmul(x, factor, y, trans = FALSE, left = TRUE, ...)

```

x                MatrixFactorization
factor           xnames(expand2(x, ...))
y
trans
left             y
...

```

facmul

```
op(M) %*% yy %*% op(M)leftMdimnamesop(M)Mt(M)trans
```

```
## Conceptually, methods for 'facmul' _would_ behave as follows ...
## Not run:
n <- 3L
x <- lu(Matrix(rnorm(n * n), n, n))
y <- rnorm(n)
L <- unname(expand2(x)[[nm <- "L"]])
stopifnot(exprs = {
  all.equal(facmul(x, nm, y, trans = FALSE, left = TRUE), L %*% y)
  all.equal(facmul(x, nm, y, trans = FALSE, left = FALSE), y %*% L)
  all.equal(facmul(x, nm, y, trans = TRUE, left = TRUE), crossprod(L, y))
  all.equal(facmul(x, nm, y, trans = TRUE, left = FALSE), tcrossprod(y, L))
})

## End(Not run)
```

fastMisc

```
as(., <class>)
```

```
.M2kind(from, kind = ".", sparse = NA)
```

```
.M2gen(from, kind = ".")
.M2sym(from, ...)
.M2tri(from, ...)
.M2diag(from)
```

```
.M2v(from)
.M2m(from)
.M2unpacked(from)
.M2packed(from)
.M2C(from)
.M2R(from)
.M2T(from)
```

```
.M2V(from)
.m2V(from, kind = ".")
```

```
.sparse2dense(from, packed = FALSE)
.diag2dense(from, kind = ".", shape = "t", packed = FALSE, uplo = "U")
.ind2dense(from, kind = "n")
.m2dense(from, class = ".ge", uplo = "U", diag = "N", trans = FALSE)
```

```
.dense2sparse(from, repr = "C")
.diag2sparse(from, kind = ".", shape = "t", repr = "C", uplo = "U")
.ind2sparse(from, kind = "n", repr = ".")
```

```
.m2sparse(from, class = ".gC", uplo = "U", diag = "N", trans = FALSE)
```

```
.tCRT(x, lazy = TRUE)
```

```
.diag.dsC(x, Chx = Cholesky(x, LDL = TRUE), res.kind = "diag")
```

```
.solve.dgC.lu (a, b, tol = .Machine$double.eps, check = TRUE)
```

```
.solve.dgC.qr (a, b, order = 3L, check = TRUE)
```

```
.solve.dgC.chol(a, b, check = TRUE)
```

```
.updateCHMfactor(object, parent, mult = 0)
```

```
fromxab      Matrix
kind          ".","n""l""d""."from","z"from"d""n"nMatrixnsparseVector
shape         ".","g""s""t""."from"g"generalMatrix
repr          ".","C""R""T""."ind2sparse"C""R"margin = 21"C"CsparseMatrix
packed        packedMatrixunpackedMatrixfromgeneralMatrix
sparse        sparseMatrixdenseMatrixNAfrom
uplo          "U""L"fromfrom
diag          "N""U"from
trans         from
class         "^[.nld](ge|sy|tr|sp|tp)".m2dense"^[.nld][gst][CRT]".m2sparse
              ".","l""d""
...           isSymmetricisTriangular
lazy
Chx           Cholesky(x, ...)xx
res.kind       c("trace", "sumLog", "prod", "min", "max", "range", "diag",
               "diagBack")
tol           lu-methods
order         qr-methods
check         dgCMatrixFALSEdgCMatrix
object        CHMfactorCholesky
parent        dsCMatrixdgCMatrix
mult
```

```
.<A>2<B>
```

```
M Matrix
V sparseVector
m
v
dense denseMatrix
unpacked unpackedMatrix
```

```

packed packedMatrix
sparse CsparseMatrixRsparseMatrixTsparseMatrix
C CsparseMatrix
R RsparseMatrix
T TsparseMatrix
gen generalMatrix
sym symmetricMatrix
tri triangularMatrix
diag diagonalMatrix
ind indMatrix

.m2dense.m2sparse.m2V

.tCRT(x)lazy = TRUE.tCRTxCRTCTlazy = FALSE.tCRTxt

.diag.dsC(x).diag.dsCChxx $LDL'$ Dres.kind

.solve.dgC.*(a, b).solve.dgC.lu(a, b)a.solve.dgC.qr(a, b)anrow(a) >= ncol(a)
.solve.dgC.chol(a, b)anrow(a) <= ncol(a)
.solve.dgC.qr.solve.dgC.chol

.updateCHMfactor(object, parent, mult).updateCHMfactorobjectF(parent) +
mult[1] * diag(nrow(parent))F(parent)mult[1]F = identityparentdsCMatrixF =
tcrossprodparentdgCMatrixF(parent)Sobject = Cholesky(S, ...)

D. <- diag(x = c(1, 1, 2, 3, 5, 8))
D.0 <- Diagonal(x = c(0, 0, 0, 3, 5, 8))
S. <- toeplitz(as.double(1:6))
C. <- new("dgCMatrix", Dim = c(3L, 4L),
          p = c(0L, 1L, 1L, 1L, 3L), i = c(1L, 0L, 2L), x = c(-8, 2, 3))

stopifnot(exprs = {
  identical(.M2tri (D.), as(D., "triangularMatrix"))
  identical(.M2sym (D.), as(D., "symmetricMatrix"))
  identical(.M2diag(D.), as(D., "diagonalMatrix"))
  identical(.M2kind(C., "l"),
            as(C., "lMatrix"))
  identical(.M2kind(.sparse2dense(C.), "l"),
            as(as(C., "denseMatrix"), "lMatrix"))
  identical(.diag2sparse(D.0, ".", "t", "C"),
            .dense2sparse(.diag2dense(D.0, ".", "t", TRUE), "C"))
  identical(.M2gen(.diag2dense(D.0, ".", "s", FALSE)),
            .sparse2dense(.M2gen(.diag2sparse(D.0, ".", "s", "T"))))
  identical(S.,
            .M2m(.m2sparse(S., ".sR"))
  identical(S. * lower.tri(S.) + diag(1, 6L),
            .M2m(.m2dense (S., ".tr", "L", "U")))
  identical(.M2R(C.), .M2R(.M2T(C.)))
  identical(.tCRT(C.), .M2R(t(C.)))
})

```

```

A <- tcrossprod(C.)/6 + Diagonal(3, 1/3); A[1,2] <- 3; A
stopifnot(exprs = {
  is.numeric( x. <- c(2.2, 0, -1.2) )
  all.equal(x., .solve.dgC.lu(A, c(1,0,0), check=FALSE))
  all.equal(x., .solve.dgC.qr(A, c(1,0,0), check=FALSE))
})

## Solving sparse least squares:

X <- rbind(A, Diagonal(3)) # design matrix X (for L.S.)
Xt <- t(X)                 # *transposed* X (for L.S.)
(y <- drop(crossprod(Xt, 1:3)) + c(-1,1)/1000) # small rand.err.
str(solveCh <- .solve.dgC.chol(Xt, y, check=FALSE)) # Xt *is* dgC..
stopifnot(exprs = {
  all.equal(solveCh$coef, 1:3, tol = 1e-3) # rel.err ~ 1e-4
  all.equal(solveCh$coef, drop(solve(tcrossprod(Xt), Xt %%% y)))
  all.equal(solveCh$coef, .solve.dgC.qr(X, y, check=FALSE))
})

```

forceSymmetric-methods

```

x symmetricMatrixas(x, "symmetricMatrix")

```

```

forceSymmetric(x, uplo)

```

```

x                matrixMatrix
uplo             "U""L""x"U"xuplosymmetricMatrixtriangularMatrixx@uplo

```

```

symmetricMatrix

```

```

symmpartas(x, <symmetricMatrix class>)

```

```

## Hilbert matrix
i <- 1:6
h6 <- 1/outer(i - 1L, i, "+")
sd <- sqrt(diag(h6))
hh <- t(h6/sd)/sd # theoretically symmetric
isSymmetric(hh, tol=0) # FALSE; hence
try( as(hh, "symmetricMatrix") ) # fails, but this works fine:
H6 <- forceSymmetric(hh)

## result can be pretty surprising:
(M <- Matrix(1:36, 6))
forceSymmetric(M) # symmetric, hence very different in lower triangle
(tm <- tril(M))
forceSymmetric(tm)

```

formatSparseM

[formatprint](#)

[formatSparseM\(\)](#)[formatSpMatrix](#)[format](#)

[.formatSparseSimple\(\)](#)

```
formatSparseM(x, zero.print = ".", align = c("fancy", "right"),
              m = as(x, "matrix"), asLogical=NULL, uniDiag=NULL,
              digits=NULL, cx, iN0, dn = dimnames(m))
```

```
.formatSparseSimple(m, asLogical=FALSE, digits=NULL,
                    col.names, note.dropping.colnames = TRUE,
                    dn=dimnames(m))
```

x	sparseMatrix
zero.print	<code>"." " " "0"</code> print()
align	zero.print formatSpMatrix
m	matrix x
asLogical	formatSparseM()
uniDiag	<code>"1" "1"</code>
digits	print.default
cx	<code>x"0.00"</code>
iN0	x
col.names	note.dropping.colnames
	formatSpMatrix
dn	dimnames NULL

cxzero.print

[formatSpMatrix](#)[formatSparseM\(\)](#)[format](#)
[printSpMatrix](#)[showprint](#)

```

m <- suppressWarnings(matrix(c(0, 3.2, 0,0, 11,0,0,0,0,-7,0), 4,9))
fm <- formatSparseM(m)
noquote(fm)
## nice, but this is nicer {with "units" vertically aligned}:
print(fm, quote=FALSE, right=TRUE)
## and "the same" as :
Matrix(m)

## align = "right" is cheaper --> the "." are not aligned:
noquote(f2 <- formatSparseM(m,align="r"))
stopifnot(f2 == fm | m == 0, dim(f2) == dim(m),
          (f2 == ".") == (m == 0))

```

generalMatrix-class

Dim, Dimnames [Matrix](#)
 factors [MatrixFactorization](#)

"Matrix"

[symmetricMatrix](#)[triangularMatrix](#)[diagonalMatrix](#)

Hilbert

nnn

Hilbert(n)

n

nn"dpoMatrix"

[dpoMatrix](#)

Hilbert(6)

image-methods

`image` $n \times mn \times m$

`image``levelplot()`"trellis"`print()`

S4 method for signature 'dgTMatrix'

```
image(x,
      xlim = c(1, di[2]),
      ylim = c(di[1], 1), aspect = "iso",
      sub = sprintf("Dimensions: %d x %d", di[1], di[2]),
      xlab = "Column", ylab = "Row", cuts = 15,
      useRaster = FALSE,
      useAbs = NULL, colorkey = !useAbs,
      col.regions = NULL,
      lwd = NULL, border.col = NULL, ...)
```

`x` `is(x, "Matrix")`

`xlimylim` `x,ylimxlim0.5ylim`

`aspect` `levelplot`

`subxlabylab` `plot.default`

`cuts`

`useRaster` `panel.levelplot.rasterlevelplotgrid.raster`

`useAbs` `abs(x)TRUEcol.regionsgreycolorkeyFALSE`

`colorkey` `useAbslistlevelplotlist(raster=TRUE)`

`col.regions` `levelplot`

`lwd` `useRasterNULLgrid.rect`

`border.col` `NANULLborder.col <- if(lwd < .01) NA else NULLNULL`
`grid::get.gpar("col")`

`...` `levelplotatcuts`

`image(<Matrix>)"trellis"levelplot()`

`dgTMatrix``showMethods(image)`

`levelplotprint.trellis`

```

showMethods(image)
## And if you want to see the method definitions:
showMethods(image, includeDefs = TRUE, inherited = FALSE)

data(CAex, package = "Matrix")
image(CAex, main = "image(CAex)") -> imgC; imgC
stopifnot(!is.null(leg <- imgC$legend), is.list(leg$right)) # failed for 2 days ..
image(CAex, useAbs=TRUE, main = "image(CAex, useAbs=TRUE)")

cCA <- Cholesky(crossprod(CAex), Imult = .01)
## See ?print.trellis --- place two image() plots side by side:
print(image(cCA, main="Cholesky(crossprod(CAex), Imult = .01)",
  split=c(x=1,y=1,nx=2, ny=1), more=TRUE)
print(image(cCA, useAbs=TRUE),
  split=c(x=2,y=1,nx=2,ny=1))

data(USCounties, package = "Matrix")
image(USCounties)# huge
image(sign(USCounties))## just the pattern
  # how the result looks, may depend heavily on
  # the device, screen resolution, antialiasing etc
  # e.g. x11(type="Xlib") may show very differently than cairo-based

## Drawing borders around each rectangle;
  # again, viewing depends very much on the device:
image(USCounties[1:400,1:200], lwd=.1)
## Using (xlim,ylim) has advantage : matrix dimension and (col/row) indices:
image(USCounties, c(1,200), c(1,400), lwd=.1)
image(USCounties, c(1,300), c(1,200), lwd=.5 )
image(USCounties, c(1,300), c(1,200), lwd=.01)
## These 3 are all equivalent :
(I1 <- image(USCounties, c(1,100), c(1,100), useAbs=FALSE))
I2 <- image(USCounties, c(1,100), c(1,100), useAbs=FALSE, border.col=NA)
I3 <- image(USCounties, c(1,100), c(1,100), useAbs=FALSE, lwd=2, border.col=NA)
stopifnot(all.equal(I1, I2, check.environment=FALSE),
  all.equal(I2, I3, check.environment=FALSE))
## using an opaque border color
image(USCounties, c(1,100), c(1,100), useAbs=FALSE, lwd=3, border.col = adjustcolor("skyblue", 1/2))

if(interactive() || nzchar(Sys.getenv("R_MATRIX_CHECK_EXTRA"))) {
  ## Using raster graphics: For PDF this would give a 77 MB file,
  ## however, for such a large matrix, this is typically considerably
  ## *slower* (than vector graphics rectangles) in most cases :
  if(doPNG <- !dev.interactive())
    png("image-USCounties-raster.png", width=3200, height=3200)
  image(USCounties, useRaster = TRUE) # should not suffer from anti-aliasing
  if(doPNG)
    dev.off()
  ## and now look at the *.png image in a viewer you can easily zoom in and out
}#only if(doExtras)

```

```
index[<-numericlogicalcharacter
```

```
[[-methods[<--methods
```

```
showClass("index")
```

```
indMatrix-class
```

```
indMatrix
```

```
pMatrixindMatrix
```

```
permmargin
```

```
RR
```

```
RpermpCpermqM
```

RM	$R \%*\% M$	$M[p,]$
MC	$M \%*\% C$	$M[, q]$
$C'M$	$\text{crossprod}(C, M)$	$M[q,]$
MR'	$\text{tcrossprod}(M, R)$	$M[, p]$
$R'R$	$\text{crossprod}(R)$	$\text{Diagonal}(x=\text{tabulate}(p, \text{ncol}(R)))$
CC'	$\text{tcrossprod}(C)$	$\text{Diagonal}(x=\text{tabulate}(q, \text{nrow}(C)))$

```
indMatrixindMatrixnsparseMatrixindMatrixdiag"logical"
```

```
new("indMatrix", ...)as(., "indMatrix")
```

```
margin
```

```
perm Dim[margin]1:Dim[1+margin%%2]
```

```
DimDimnames Matrix
```

```
"sparseMatrix""generalMatrix"
```

```

%% signature(x = "indMatrix", y = "Matrix")showMethods("%*%", classes =
  "indMatrix")
coerce signature(from = "numeric", to = "indMatrix")indMatrix
coerce signature(from = "list", to = "indMatrix")indMatrix
coerce signature(from = "indMatrix", to = "matrix")matrixFALSETRUE
t signature(x = "indMatrix")indMatrixpermmargin
rowSumsrowMeanscolSumscolMeans signature(x = "indMatrix")
rbind2cbind2 signature(x = "indMatrix", y = "indMatrix")
  signature(X = "indMatrix", Y = "indMatrix")

uni-muenchen.depMatrixcrossprod(x, y)kronecker(x, y)indMatrixmarginindMatrix

```

pMatrixnMatrix

```

p1 <- as(c(2,3,1), "pMatrix")
(sm1 <- as(rep(c(2,3,1), e=3), "indMatrix"))
stopifnot(all(sm1 == p1[rep(1:3, each=3),]))

## row-indexing of a <pMatrix> turns it into an <indMatrix>:
class(p1[rep(1:3, each=3),])

set.seed(12) # so we know '10' is in sample
## random index matrix for 30 observations and 10 unique values:
(s10 <- as(sample(10, 30, replace=TRUE),"indMatrix"))

## Sample rows of a numeric matrix :
(mm <- matrix(1:10, nrow=10, ncol=3))
s10 %*% mm

set.seed(27)
IM1 <- as(sample(1:20, 100, replace=TRUE), "indMatrix")
IM2 <- as(sample(1:18, 100, replace=TRUE), "indMatrix")
(c12 <- crossprod(IM1,IM2))
## same as cross-tabulation of the two index vectors:
stopifnot(all(c12 - unclass(table(IM1@perm, IM2@perm)) == 0))

# 3 observations, 4 implied values, first does not occur in sample:
as(2:4, "indMatrix")
# 3 observations, 5 values, first and last do not occur in sample:
as(list(2:4, 5), "indMatrix")

as(sm1, "nMatrix")
s10[1:7, 1:4] # gives an "ngTMatrix" (most economic!)
s10[1:4, ] # preserves "indMatrix"-class

I1 <- as(c(5:1,6:4,7:3), "indMatrix")
I2 <- as(7:1, "pMatrix")
(I12 <- rbind(I1, I2))

```

```
stopifnot(is(I12, "indMatrix"),
          identical(I12, rbind(I1, I2)),
          colSums(I12) == c(2L, 2:4, 4:2))
```

invertPerm

```
invertPermsignPermnisPermnasPermmnm <= n
```

```
invertPerm(p, off = 1L, ioff = 1L)
signPerm(p, off = 1L)
isPerm(p, off = 1L)
asPerm(pivot, off = 1L, ioff = 1L, n = length(pivot))
```

```
invPerm(p, zero.p = FALSE, zero.res = FALSE)
```

p	n
pivot	m
off	poff+0:(n-1)pivotmoff+0:(n-1)
ioff	ioff+0:(n-1)
n	mseq_len(n)
zero.p	off=0TRUEoff=1FALSE
zero.res	ioff=0TRUEioff=1FALSE

```
invertPerm(p, off, ioff=1)order(p)sort.list(p)offoff=1pp[p] <- seq_along(p)
invPerminvertPerm
```

```
off=1ioff=1
invertPerm(p)length(p)p[invertPerm(p)]invertPerm(p)[p]seq_along(p)
signPerm(p)p-1p
isPerm(p)TRUEpseq_along(p)FALSE
asPerm(pivot)ipivot[i]seq_len(n)iseq_along(pivot)
```

[pMatrix](#)

```

p <- sample(10L) # a random permutation vector
ip <- invertPerm(p)
s <- signPerm(p)

## 'p' and 'ip' are indeed inverses:
stopifnot(exprs = {
  isPerm(p)
  isPerm(ip)
  identical(s, 1L) || identical(s, -1L)
  identical(s, signPerm(ip))
  identical(p[ip], 1:10)
  identical(ip[p], 1:10)
  identical(invertPerm(ip), p)
})

## Product of transpositions (1 2)(2 1)(4 3)(6 8)(10 1) = (3 4)(6 8)(1 10)
pivot <- c(2L, 1L, 3L, 3L, 5L, 8L, 7L, 8L, 9L, 1L)
q <- asPerm(pivot)
stopifnot(exprs = {
  identical(q, c(10L, 2L, 4L, 3L, 5L, 8L, 7L, 6L, 9L, 1L))
  identical(q[q], seq_len(10L)) # because the permutation is odd:
  signPerm(q) == -1L
})

invPerm # a less general version of 'invertPerm'

```

is.na-methods

[anyNA\(\)](#)[is.na\(\)](#)[is.nan\(\)](#)[is.infinite\(\)](#)[is.finite\(\)](#)[Matrix](#)[sparseVector](#)

```

## S4 method for signature 'denseMatrix'
is.na(x)
## S4 method for signature 'sparseMatrix'
is.na(x)
## S4 method for signature 'diagonalMatrix'
is.na(x)
## S4 method for signature 'indMatrix'
is.na(x)
## S4 method for signature 'sparseVector'
is.na(x)
## ...
## and likewise for anyNA, is.nan, is.infinite, is.finite

```

x

[is.*\(\)](#)[nMatrix](#)[sparseVector](#)[xx](#)[NANaInf](#)-Inf[anyNA](#)[TRUE](#)[xNANa](#)[FALSE](#)

NANaNInf

```
(M <- Matrix(1:6, nrow = 4, ncol = 3,
            dimnames = list(letters[1:4], LETTERS[1:3])))
stopifnot(!anyNA(M), !any(is.na(M)))

M[2:3, 2] <- NA
(inM <- is.na(M))
stopifnot(anyNA(M), sum(inM) == 2)

(A <- spMatrix(nrow = 10, ncol = 20,
              i = c(1, 3:8), j = c(2, 9, 6:10), x = 7 * (1:7)))
stopifnot(!anyNA(A), !any(is.na(A)))

A[2, 3] <- A[1, 2] <- A[5, 5:9] <- NA
(inA <- is.na(A))
stopifnot(anyNA(A), sum(inA) == 1 + 1 + 5)
```

is.null.DN

dn

dimnamesdnNULL

is.null.DN(dn)is.null(dn)TRUEdnNULLlist(NULL,NULL)matrixclassdimnames

is.null.DN(dn)

dn

dimnames()matrix

logicalTRUEFALSE

Matrixlist(NULL,NULL)

is.null.dimnames.matrix

```

m1 <- m2 <- m3 <- m4 <- m <-
  matrix(round(100 * rnorm(6)), 2, 3)
dimnames(m1) <- list(NULL, NULL)
dimnames(m2) <- list(NULL, character())
dimnames(m3) <- rev(dimnames(m2))
dimnames(m4) <- rep(list(character()),2)

m4 # prints absolutely identically to m

c.o <- capture.output
cm <- c.o(m)
stopifnot(exprs = {
  m == m1; m == m2; m == m3; m == m4
  identical(cm, c.o(m1)); identical(cm, c.o(m2))
  identical(cm, c.o(m3)); identical(cm, c.o(m4))
})

hasNoDimnames <- function(.) is.null.DN(dimnames(.))
stopifnot(exprs = {
  hasNoDimnames(m)
  hasNoDimnames(m1); hasNoDimnames(m2)
  hasNoDimnames(m3); hasNoDimnames(m4)
  hasNoDimnames(Matrix(m) -> M)
  hasNoDimnames(as(M, "sparseMatrix"))
})

```

isSymmetric-methods

```

isSymmetric[dD]imnamesisSymmetricclass"matrix"isSymmetric"Matrix"

```

```

## S4 method for signature 'denseMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'CsparseMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'RsparseMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'TsparseMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'diagonalMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'indMatrix'
isSymmetric(object, checkDN = TRUE, ...)
## S4 method for signature 'dgeMatrix'
isSymmetric(object, checkDN = TRUE, tol = 100 * .Machine$double.eps, tol1 = 8 * tol, ...)
## S4 method for signature 'dgCMatrix'
isSymmetric(object, checkDN = TRUE, tol = 100 * .Machine$double.eps, ...)

```

```

object      "Matrix"
checkDN     Dimnamesobject
toltol1     isSymmetric.matrix
...         all.equal

```

```
Dimnamesobjectdn
```

```

dn[[1]]dn[[2]]NULL
ndn <- names(dn)NULLndn[1]ndn[2]"

```

```

list(a=nms, a=nms)list(a=nms, NULL)list(NULL, a=nms)
isSymmetric.matrixdn[1]dn[2]dndimnames

```

```
TRUEFALSENA
```

```
forceSymmetricssympartskewpart"symmetricMatrix"
```

```

isSymmetric(Diagonal(4)) # TRUE of course
M <- Matrix(c(1,2,2,1), 2,2)
isSymmetric(M) # TRUE (*and* of formal class "dsyMatrix")
isSymmetric(as(M, "generalMatrix")) # still symmetric, even if not "formally"
isSymmetric(triu(M)) # FALSE

## Look at implementations:
showMethods("isSymmetric", includeDefs = TRUE) # includes S3 generic from base

```

```
isTriangular-methods
```

```
isTriangularisDiagonalisSymmetricbase"matrix""Matrix"
```

```
isTriangular(object, upper = NA, ...)
```

```
isDiagonal(object)
```

```

object
upper      TRUEFALSETRUEobjectNATRUEobject
...

```

```
TRUEFALSENA
objectupperNAisTriangularTRUEkind"U""L"objectkind
```

```
isSymmetric"triangularMatrix""diagonalMatrix"
```

```
isTriangular(Diagonal(4))
## is TRUE: a diagonal matrix is also (both upper and lower) triangular
(M <- Matrix(c(1,2,0,1), 2,2))
isTriangular(M) # TRUE (*and* of formal class "dtrMatrix")
isTriangular(as(M, "generalMatrix")) # still triangular, even if not "formally"
isTriangular(crossprod(M)) # FALSE

isDiagonal(matrix(c(2,0,0,1), 2,2)) # TRUE

## Look at implementations:
showMethods("isTriangular", includeDefs = TRUE)
showMethods("isDiagonal", includeDefs = TRUE)
```

KhatriRao

```
KhatriRao(X, Y = X, FUN = "*", sparseY = TRUE, make.dimnames = FALSE)
```

```
XY
FUN          functionkronecker
sparseY      YsparseMatrixFALSE
make.dimnames dimnamesXY
```

```
"CsparseMatrix"RXn × kYm × k(n · m) × kR[,j]kronecker(X[,j], Y[,j])
```

kronecker

```
## Example with very small matrices:
m <- matrix(1:12,3,4)
d <- diag(1:4)
KhatriRao(m,d)
KhatriRao(d,m)
dimnames(m) <- list(LETTERS[1:3], letters[1:4])
KhatriRao(m,d, make.dimnames=TRUE)
KhatriRao(d,m, make.dimnames=TRUE)
dimnames(d) <- list(NULL, paste0("D", 1:4))
KhatriRao(m,d, make.dimnames=TRUE)
KhatriRao(d,m, make.dimnames=TRUE)
dimnames(d) <- list(paste0("d", 10*1:4), paste0("D", 1:4))
(Kmd <- KhatriRao(m,d, make.dimnames=TRUE))
(Kdm <- KhatriRao(d,m, make.dimnames=TRUE))

nm <- as(m, "nsparseMatrix")
nd <- as(d, "nsparseMatrix")
KhatriRao(nm,nd, make.dimnames=TRUE)
KhatriRao(nd,nm, make.dimnames=TRUE)

stopifnot(dim(KhatriRao(m,d)) == c(nrow(m)*nrow(d), ncol(d)))
## border cases / checks:
zm <- nm; zm[] <- FALSE # all FALSE matrix
stopifnot(all(K1 <- KhatriRao(nd, zm) == 0), identical(dim(K1), c(12L, 4L)),
           all(K2 <- KhatriRao(zm, nd) == 0), identical(dim(K2), c(12L, 4L)))

d0 <- d; d0[] <- 0; m0 <- Matrix(d0[-1,])
stopifnot(all(K3 <- KhatriRao(d0, m) == 0), identical(dim(K3), dim(Kdm)),
           all(K4 <- KhatriRao(m, d0) == 0), identical(dim(K4), dim(Kmd)),
           all(KhatriRao(d0, d0) == 0), all(KhatriRao(m0, d0) == 0),
           all(KhatriRao(d0, m0) == 0), all(KhatriRao(m0, m0) == 0),
           identical(dimnames(KhatriRao(m, d0, make.dimnames=TRUE)), dimnames(Kmd)))

## a matrix with "structural" and non-structural zeros:
m01 <- new("dgCMatrix", i = c(0L, 2L, 0L, 1L), p = c(0L, 0L, 0L, 2L, 4L),
          Dim = 3:4, x = c(1, 0, 1, 0))
D4 <- Diagonal(4, x=1:4) # "as" d
DU <- Diagonal(4)# unit-diagonal: uplo="U"
(K5 <- KhatriRao( d, m01))
K5d <- KhatriRao( d, m01, sparseY=FALSE)
K5Dd <- KhatriRao(D4, m01, sparseY=FALSE)
K5Ud <- KhatriRao(DU, m01, sparseY=FALSE)
(K6 <- KhatriRao(diag(3), t(m01)))
K6D <- KhatriRao(Diagonal(3), t(m01))
K6d <- KhatriRao(diag(3), t(m01), sparseY=FALSE)
K6Dd <- KhatriRao(Diagonal(3), t(m01), sparseY=FALSE)
stopifnot(exprs = {
  all(K5 == K5d)
  identical(cbind(c(7L, 10L), c(3L, 4L)),
            which(K5 != 0, arr.ind = TRUE, useNames=FALSE))
  identical(K5d, K5Dd)
  identical(K6, K6D)
```

```

    all(K6 == K6d)
    identical(cbind(3:4, 1L),
              which(K6 != 0, arr.ind = TRUE, useNames=FALSE))
    identical(K6d, K6Dd)
  })

```

KNex

```
mmymm"dgCMatrix"ynumeric
```

```
data(KNex)
```

```

data(KNex, package = "Matrix")
class(KNex$mm)
dim(KNex$mm)
image(KNex$mm)
str(KNex)

system.time( # a fraction of a second
  sparse.sol <- with(KNex, solve(crossprod(mm), crossprod(mm, y))))

head(round(sparse.sol,3))

## Compare with QR-based solution ("more accurate, but slightly slower"):
system.time(
  sp.sol2 <- with(KNex, qr.coef(qr(mm), y) ))

all.equal(sparse.sol, sp.sol2, tolerance = 1e-13) # TRUE

```

kronecker-methods

```
"Matrix"
```

```
0 * NA0NAkronecker
```

```

signature(X = "Matrix", Y = "ANY")
signature(X = "ANY", Y = "Matrix")
signature(X = "diagonalMatrix", Y = "ANY")
signature(X = "sparseMatrix", Y = "ANY")
signature(X = "TsparseMatrix", Y = "TsparseMatrix")
signature(X = "dgTMatrix", Y = "dgTMatrix")
signature(X = "dtTMatrix", Y = "dtTMatrix")
signature(X = "indMatrix", Y = "indMatrix")

```

```

(t1 <- spMatrix(5,4, x= c(3,2,-7,11), i= 1:4, j=4:1)) # 5 x 4
(t2 <- kronecker(Diagonal(3, 2:4), t1)) # 15 x 12

## should also work with special-cased logical matrices
l3 <- upper.tri(matrix(,3,3))
M <- Matrix(l3)
(N <- as(M, "nsparseMatrix")) # "ntCMatrix" (upper triangular)
N2 <- as(N, "generalMatrix") # (lost "t"riangularity)
MM <- kronecker(M,M)
NN <- kronecker(N,N) # "dtTMatrix" i.e. did keep
NN2 <- kronecker(N2,N2)
stopifnot(identical(NN,MM),
           is(NN2, "sparseMatrix"), all(NN2 == NN),
           is(NN, "triangularMatrix"))

```

ldenseMatrix-class

ldenseMatrixdenseMatrixlMatrix

x

DimDimnames Matrix

"lMatrix""denseMatrix""Matrix""lMatrix""Matrix""denseMatrix"

signature(x = "ldenseMatrix", mode = "missing")

signature(x = "ndenseMatrix")which(x, arr.ind)lMatrix

lgeMatrix

showClass("ldenseMatrix")

as(diag(3) > 0, "ldenseMatrix")

ldiMatrix-class

```
"ldiMatrix"

new("ldiMatrix", ...)Diagonal

x "logical"
diag "character"ddiMatrix
DimDimnames dimnamesMatrix

"diagonalMatrix""lMatrix"
"sparseMatrix""diagonalMatrix"

ddiMatrixdiagonalMatrixDiagonal

(lM <- Diagonal(x = c(TRUE,FALSE,FALSE)))
str(lM)#> gory details (slots)

crossprod(lM) # numeric
(nM <- as(lM, "nMatrix"))
crossprod(nM) # pattern sparse
```

lgeMatrix-class

```
logical

x "logical"
DimDimnames "integer"NULLMatrix
factors "list"

"ldenseMatrix""lMatrix""ldenseMatrix""denseMatrix""ldenseMatrix""Matrix"
"ldenseMatrix""Matrix""ldenseMatrix"

t()as(.)showMethods(class="lgeMatrix")
```


[ltrMatrix](#)[lsyMatrix](#)[lgCMatrix](#)

```
showClass("lgeMatrix")
str(new("lgeMatrix"))
set.seed(1)
(lM <- Matrix(matrix(rnorm(28), 4,7) > 0))# a simple random lgeMatrix
set.seed(11)
(lC <- Matrix(matrix(rnorm(28), 4,7) > 0))# a simple random lgCMatrix
as(lM, "CsparseMatrix")
```

`lsparseMatrix`-class

`lsparseMatrix`[TRUE](#)[FALSE](#)[NA](#)

[TsparseMatrix](#)[lgTMatrix](#)[lsTMatrix](#)[ltTMatrix](#)[CsparseMatrix](#)[lgCMatrix](#)[lsCMatrix](#)
[ltCMatrix](#)[RsparseMatrix](#)[lgRMatrix](#)[lsRMatrix](#)[ltRMatrix](#)[xgst](#)

[TsparseMatrix](#)[lgTMatrix](#)(*i,j*)[dgTMatrix](#)*xx*(*i,j*)*or*[TRUE](#) + [TRUE](#) |-> [TRUE](#)[TRUE](#) + [FALSE](#) |->
[TRUE](#)[asUniqueT](#)()(*i,j*)

`new("lgCMatrix", ...)`*x* != 0

x "logical"[TRUE](#)[NA](#)[FALSE](#)
uplo "character"
diag "character""U""N"*diag*"U"
p "integer"
i "integer"
j "integer"
Dim "integer"

`signature`(*from* = "dgCMatrix", *to* = "lgCMatrix")
`signature`(*x* = "lgCMatrix")*x*
`signature`(*x* = "lsparseMatrix")[which](#)(*x*, *arr.ind*)[lMatrix](#)

[dgCMatrix](#)[dgTMatrix](#)

```

(m <- Matrix(c(0,0,2:0), 3,5, dimnames=list(LETTERS[1:3],NULL)))
(lm <- (m > 1)) # lgC
!lm      # no longer sparse
stopifnot(is(lm,"lsparseMatrix"),
           identical(!lm, m <= 1))

data(KNex, package = "Matrix")
str(mmG.1 <- (KNex $ mm) > 0.1)# "lgC..."
table(mmG.1@x)# however with many ``non-structural zeros''
## from logical to nz_pattern -- okay when there are no NA's :
nmG.1 <- as(mmG.1, "nMatrix") # <<< has "TRUE" also where mmG.1 had FALSE
## from logical to "double"
dmG.1 <- as(mmG.1, "dMatrix") # has '0' and back:
lmG.1 <- as(dmG.1, "lMatrix")
stopifnot(identical(nmG.1, as((KNex $ mm) != 0,"nMatrix")),
           validObject(lmG.1),
           identical(lmG.1, mmG.1))

class(xnx <- crossprod(nmG.1))# "nsC.."
class(xlx <- crossprod(mmG.1))# "dsC.." : numeric
is0 <- (xlx == 0)
mean(as.vector(is0))# 99.3% zeros: quite sparse, but
table(xlx@x == 0)# more than half of the entries are (non-structural!) 0
stopifnot(isSymmetric(xlx), isSymmetric(xnx),
           ## compare xnx and xlx : have the *same* non-structural 0s :
           sapply(slotNames(xnx),
                   function(n) identical(slot(xnx, n), slot(xlx, n))))

```

lsyMatrix-class

"lsyMatrix""lspMatrix"

new("lsyMatrix", ...)

uplo "character"

x "logical"

DimDimnames "integer"NULLMatrix

factors "list"

"ldenseMatrix""symmetricMatrix""Matrix"showClass("lsyMatrix")

t()as(.)showMethods(class="lsyMatrix")

lgeMatrixMatrixt

```

(M2 <- Matrix(c(TRUE, NA, FALSE, FALSE), 2, 2)) # logical dense (ltr)
str(M2)
# can
(sM <- M2 | t(M2)) # "lge"
as(sM, "symmetricMatrix")
str(sM <- as(sM, "packedMatrix")) # packed symmetric

```

ltrMatrix-class

```
"ltrMatrix""ltpMatrix"
```

```

x "logical"
uplo "character"
diag "character""U""N"triangularMatrix
DimDimnames "integer"NULLMatrix
factors "list"

```

```
"ldenseMatrix""triangularMatrix""Matrix""lMatrix"showClass("ltrMatrix")
```

```
t()as\(.\)showMethods(class="ltrMatrix")
```

```
lgeMatrixMatrixt
```

```

showClass("ltrMatrix")

str(new("ltpMatrix"))
(lutr <- as(upper.tri(matrix(, 4, 4)), "ldenseMatrix"))
str(lutp <- pack(lutr)) # packed matrix: only 10 = 4*(4+1)/2 entries
!!lutp # the logical negation (is not logical triangular !)
## but this one is:
stopifnot(all.equal(lutp, pack(!lutp)))

```

lu-methods

$m \times n$ A

$$P_1 A P_2 = LU$$

$$A = P_1' L U P_2'$$

$P_1 m \times m P_2 n \times n L m \times m \min(m, n) U \min(m, n) \times n$

`denseMatrix``dgetrf` P_2

`sparseMatrix``cs_lum` $= nLU$

```
lu(x, ...)
## S4 method for signature 'dgeMatrix'
lu(x, warnSing = TRUE, ...)
## S4 method for signature 'dgCMatrix'
lu(x, errSing = TRUE, order = NA_integer_,
    tol = 1, ...)
## S4 method for signature 'dsyMatrix'
lu(x, cache = TRUE, ...)
## S4 method for signature 'dsCMatrix'
lu(x, cache = TRUE, ...)
## S4 method for signature 'matrix'
lu(x, ...)
```

```
x          Matrix
warnSing    x
errSing     x
order       0:3cs_sqr $P_2 A + A' \tilde{A}' \tilde{A} A' A \tilde{A} A N$ atol == 1A
tol         tol * aatol < 1
cache       x@factors[["LU"]]generalMatrixsymmetricMatrix
...
```

```
xdgeMatrixwarnSing = TRUEdenseLUtryCatchdgCMatrixerrSing = TRUENaxerrSing =
FALSEis(., "sparseLU")
```

`LU``denseLU``x``sparseMatrix``sparseLU`

<https://netlib.org/lapack/double/dgetrf.f>

[denseLU](#)[sparseLU](#)
[dgeMatrix](#)[dgCMatrix](#)
[expand1](#)[expand2](#)
[Cholesky](#)[BunchKaufman](#)[Schur](#)[qr](#)

```
showMethods("lu", inherited = FALSE)
set.seed(0)
```

```
## ---- Dense -----

(A1 <- Matrix(rnorm(9L), 3L, 3L))
(lu.A1 <- lu(A1))

(A2 <- round(10 * A1[, -3L]))
(lu.A2 <- lu(A2))

## A ~ P1' L U in floating point
str(e.lu.A2 <- expand2(lu.A2), max.level = 2L)
stopifnot(all.equal(A2, Reduce(`%*%`, e.lu.A2)))

## ---- Sparse -----

A3 <- as(readMM(system.file("external/pores_1.mtx", package = "Matrix")),
         "CsparseMatrix")
(lu.A3 <- lu(A3))

## A ~ P1' L U P2' in floating point
str(e.lu.A3 <- expand2(lu.A3), max.level = 2L)
stopifnot(all.equal(A3, Reduce(`%*%`, e.lu.A3)))
```

`mat2triplet`

["TsparseMatrix"](#)["nMatrix"](#)

```
mat2triplet(x, uniqT = FALSE)
```

```
x          as(x, "TsparseMatrix")matrix
uniqT      logicalasUniqueT(byrow=FALSE)
```

[list](#)

```
i          x
i          x
x          xas(x, "TsparseMatrix")"nsparseMatrix"
order"TsparseMatrix""jij"
```

```
mat2triplet(summary(<sparseMatrix>))ij"diagonalMatrix"
```

```
summary("sparseMatrix")summary,sparseMatrix-method  
mat2triplet()spMatrixsparseMatrix
```

```
mat2triplet # simple definition  
  
i <- c(1,3:8); j <- c(2,9,6:10); x <- 7 * (1:7)  
(Ax <- sparseMatrix(i, j, x = x)) ## 8 x 10 "dgCMatrix"  
str(trA <- mat2triplet(Ax))  
stopifnot(i == sort(trA$i), sort(j) == trA$j, x == sort(trA$x))  
  
D <- Diagonal(x=4:2)  
summary(D)  
str(mat2triplet(D))
```

matmult-methods

```
%*%MatrixsparseVectormatrix  
crossprodtcrossprodt(.)symmetricMatrixcrossprod(m)  
tcrossprod()tcrossprod(x)x %*% t(x)tcrossprod(x, y)x %*% t(y)  
%%&%boolArith = TRUE  
  
## S4 method for signature 'CsparseMatrix,diagonalMatrix'  
x %*% y  
  
## S4 method for signature 'CsparseMatrix,diagonalMatrix'  
crossprod(x, y = NULL, boolArith = NA, ...)  
## .... and for many more signatures  
  
## S4 method for signature 'TsparseMatrix,missing'  
tcrossprod(x, y = NULL, boolArith = NA, ...)  
## .... and for many more signatures  
  
x  
y [t]crossprod()NULLy = x  
boolArith logicalNATRUEFALSE"nMatrix"NA"lMatrix"NAxy"nsparseMatrix"  
...  
  
MatrixdgCMatrix  
boolArith = TRUE%%&%&%
```

MatrixsymmetricMatrix

```
signature(x = "dgeMatrix", y = "dgeMatrix")showMethods("%*%", class =
"dgeMatrix")
signature(x = "dtrMatrix", y = "matrix")showMethods("%*%", class="dtrMatrix")
signature(x = "dgeMatrix", y = "dgeMatrix")showMethods("crossprod", class =
"dgeMatrix")t(x) %*% y
signature(x = "CsparseMatrix", y = "missing")t(x) %*% xdsCMatrix
signature(x = "TsparseMatrix", y = "missing")t(x) %*% xdsCMatrix
signature(x = "dtrMatrix", y = "matrix")"%*%"
```

```
boolArith = TRUEFALSENAx%%
boolArith = TRUECsparseMatrix
```

tcrossprodcrossprod*%%&%

```
## A random sparse "incidence" matrix :
m <- matrix(0, 400, 500)
set.seed(12)
m[runif(314, 0, length(m))] <- 1
mm <- as(m, "CsparseMatrix")
object.size(m) / object.size(mm) # smaller by a factor of > 200

## tcrossprod() is very fast:
system.time(tCmm <- tcrossprod(mm))# 0 (PIII, 933 MHz)
system.time(cm <- crossprod(t(m))) # 0.16
system.time(cm. <- tcrossprod(m)) # 0.02

stopifnot(cm == as(tCmm, "matrix"))

## show sparse sub matrix
tCmm[1:16, 1:30]
```

Matrix

Matrix

```
Matrix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL,
       sparse = NULL, doDiag = TRUE, forceCheck = FALSE)
```

```

data
nrow      data
ncol      data
byrow     FALSE
dimnames  dimnameslistNULL
sparse    NULL
doDiag    diagonalMatrixdiagonalMatrixextendssparseMatrixsparse
          doDiagsparse
forceCheck data"Matrix"

```

```

nrowncoldataMatrix()logicalMatrixdimnamesdimnames = list(NULL,NULL)
matrixMatrix(0, nrow,ncol)Matrix(FALSE, *)
matrix"Matrix""Matrix"

```

```

"Matrix"datamatrixMatrixnrowncolbyrow

```

```

MatrixsymmetricMatrixtriangularMatrixdiagonalMatrixmatrix
sparseMatrixbdiagbandSparseDiagonal

```

```

Matrix(0, 3, 2)          # 3 by 2 matrix of zeros -> sparse
Matrix(0, 3, 2, sparse=FALSE)# -> 'dense'

## 4 cases - 3 different results :
Matrix(0, 2, 2)          # diagonal !
Matrix(0, 2, 2, sparse=FALSE)# (ditto)
Matrix(0, 2, 2,          doDiag=FALSE)# -> sparse symm. "dsCMatrix"
Matrix(0, 2, 2, sparse=FALSE, doDiag=FALSE)# -> dense symm. "dsyMatrix"

Matrix(1:6, 3, 2)        # a 3 by 2 matrix (+ integer warning)
Matrix(1:6 + 1, nrow=3)

## logical ones:
Matrix(diag(4) > 0) # -> "ldiMatrix" with diag = "U"
Matrix(diag(4) > 0, sparse=TRUE) # (ditto)
Matrix(diag(4) >= 0) # -> "lsyMatrix" (of all 'TRUE')
## triangular
l3 <- upper.tri(matrix(,3,3))
(M <- Matrix(l3)) # -> "ltCMatrix"
Matrix(! l3)      # -> "ltrMatrix"
as(l3, "CsparseMatrix")# "lgCMatrix"

Matrix(1:9, nrow=3,
      dimnames = list(c("a", "b", "c"), c("A", "B", "C")))
(I3 <- Matrix(diag(3)))# identity, i.e., unit "diagonalMatrix"
str(I3) # note 'diag = "U"' and the empty 'x' slot

```



```
(A <- cbind(a=c(2,1), b=1:2))# symmetric *apart* from dimnames
Matrix(A) # hence 'dgeMatrix'
(As <- Matrix(A, dimnames = list(NULL,NULL)))# -> symmetric
forceSymmetric(A) # also symmetric, w/ symm. dimnames
stopifnot(is(As, "symmetricMatrix"),
           is(Matrix(0, 3,3), "sparseMatrix"),
           is(Matrix(FALSE, 1,1), "sparseMatrix"))
```

Matrix-class

Matrix

Dim

Dimnames NULLDim

```
signature(x = "Matrix", logarithm = "missing")
signature(x = "Matrix", logarithm = "logical")logxxdetdeterminant()Matrix::det
base::detdeterminant()
signature(x = "Matrix")diff()diff()
signature(x = "Matrix")dim
signature(x = "Matrix", value = "ANY")valueprod(value) == prod(dim(x))
signature(x = "Matrix")dimnames
signature(x = "Matrix", value = "list")dimnameslistdimnames<-
signature(x = "Matrix")prod(dim(x))"double"
signature(object = "Matrix")showprintprintSpMatrix
signature(x = "Matrix")"dMatrix"round()
signature(object = "Matrix")imagelevelplot()
signature(object = "Matrix")
signature(object = "Matrix")

signature(x = "Matrix")as(x, "matrix")
signature(x = "Matrix", mode = "missing")as.vector(m)as.vector(as(m, "matrix"))

signature(from = "ANY", to = "Matrix")as.matrix()from
"Matrix"colSumsrowMeansas.matrix()applyeigensvdkappamatrix

Matrixas.matrixas.arrayfunction(x) as(x, "matrix")as.matrix(m)as.array(m)m
"Matrix"applyouter"Matrix"
```

<bates@stat.wisc.edu>

[dgeMatrix](#)[dgCMatrix](#)[Matrix](#)
[kronecker](#)

```
slotNames("Matrix")

cl <- getClass("Matrix")
names(cl@subclasses) # more than 40 ..

showClass("Matrix")#> output with slots and all subclasses

(M <- Matrix(c(0,1,0,0), 6, 4))
dim(M)
diag(M)
cm <- M[1:4,] + 10*Diagonal(4)
diff(M)
## can reshape it even :
dim(M) <- c(2, 12)
M
stopifnot(identical(M, Matrix(c(0,1,0,0), 2,12)),
           all.equal(det(cm),
                     determinant(as(cm,"matrix"), log=FALSE)$modulus,
                     check.attributes=FALSE))
```

Matrix-notyet

[iMatrix](#)[Matrix](#)
[zMatrix](#)[complexMatrix](#)

```
showClass("iMatrix")
showClass("zMatrix")
```

MatrixClass

[classclcharacter](#)(0)

```
MatrixClass(cl, cld = getClassDef(cl), ...Matrix = TRUE,
            dropVirtual = TRUE, ...)
```

```

cl
cld
...Matrix      logical"[dlniz]..Matrix"
dropVirtual    logical
...            .selectSuperClasses()

character

```

Matrix

```

mkA <- setClass("A", contains="dgCMatrix")
(A <- mkA())
stopifnot(identical(
  MatrixClass("A"),
  "dgCMatrix"))

```

MatrixFactorization-class

MatrixFactorization $m \times n$ A

$$P_1 A P_2 = A_1 \cdots A_p$$

$$A = P_1' A_1 \cdots A_p P_2'$$

$$P_1 P_2 A P_2 = P_1' P_1 = I_m P_2 = I_n I_m I_n m \times mn \times n$$

CholeskyFactorizationBunchKaufmanFactorizationSchurFactorizationLUQR
MatrixFactorization

Dim

Dimnames dimnamesNULLDim

```

determinant signature(x = "MatrixFactorization", logarithm = "missing")logarithm
= TRUE
dim signature(x = "MatrixFactorization")x@Dim
dimnames signature(x = "MatrixFactorization")x@Dimnames
dimnames<- signature(x = "MatrixFactorization", value = "NULL")xx@Dimnames
list(NULL, NULL)
dimnames<- signature(x = "MatrixFactorization", value = "list")xx@Dimnamesvalue
length signature(x = "MatrixFactorization")prod(x@Dim)
show signature(object = "MatrixFactorization")str
solve signature(a = "MatrixFactorization", b = .)solve-methods
unname signature(obj = "MatrixFactorization")objobj@Dimnameslist(NULL, NULL)

```

CholeskyFactorizationCholeskypCholeskyCHMfactor
BunchKaufmanFactorizationBunchKaufmanpBunchKaufman
SchurFactorizationSchur
LUdenseLUsparseLU
QRsparseQR
CholeskyBunchKaufmanSchurluqr
expand1expand2MatrixFactorization

showClass("MatrixFactorization")

ndenseMatrix-class

ndenseMatrixdenseMatrixlMatrix

x

DimDimnames Matrix

"nMatrix""denseMatrix""Matrix""nMatrix""Matrix""denseMatrix"

```
signature(x = "nsparseMatrix", y = "ndenseMatrix")
signature(x = "ndenseMatrix", y = "nsparseMatrix")
signature(x = "nsparseMatrix", y = "ndenseMatrix")
signature(x = "ndenseMatrix", y = "nsparseMatrix")
signature(x = "ndenseMatrix", mode = "missing")
signature(x = "ndenseMatrix")diag()
signature(x = "ndenseMatrix")which(x, arr.ind)lMatrix
```

ngeMatrix

showClass("ndenseMatrix")

as(diag(3) > 0, "ndenseMatrix")# -> "nge"

nearPD

```
nearPD(x, corr = FALSE, keepDiag = FALSE, base.matrix = FALSE,
       do2eigen = TRUE, doSym = FALSE,
       doDykstra = TRUE, only.values = FALSE,
       ensureSymmetry = !isSymmetric(x),
       eig.tol = 1e-06, conv.tol = 1e-07, posd.tol = 1e-08,
       maxit = 100, conv.norm.type = "I", trace = FALSE)
```

```
x           $n \times n$ ensureSymmetry  
corr  
keepDiag    corrTRUEdiag(x)  
base.matrix matmatrixMatrixdpoMatrix  
do2eigen    posdefify()  
doSym        $X \leftarrow (X + t(X))/2$   $X \leftarrow \text{tcrossprod}(Qd, Q)$   
doDykstra    $Y_k = P_U(P_S(Y_{k-1}))$   
only.values TRUE  
ensureSymmetry symmpart(x)isSymmetric(x)TRUEFALSEFALSEx  
eig.tol       $\lambda_1 \lambda_k \lambda_k / \lambda_1 \leq \text{eig.tol}$   
conv.tol  
posd.tol     posdefifydo2eigenTRUE  
maxit  
conv.norm.type norm(*, type)"I""F"  
trace
```

```
do2eigenposdefify  
corr = TRUEdiag(.) <- 1  
doDykstra = FALSEnearPD()
```

```
only.values = TRUEclass"nearPD"  
mat          dpoMatrix  
eigenvalues  mat  
corr         corr  
normF        norm(x-X, "F")  
iterations  
converged
```

```
corr=TRUEnearcor()posdefify()
```

```
## Higham(2002), p.334f - simple example
A <- matrix(1, 3,3); A[1,3] <- A[3,1] <- 0
n.A <- nearPD(A, corr=TRUE, do2eigen=FALSE)
n.A[c("mat", "normF")]
n.A.m <- nearPD(A, corr=TRUE, do2eigen=FALSE, base.matrix=TRUE)$mat
stopifnot(exprs = {                                     #=-----
  all.equal(n.A$mat[1,2], 0.760689917)
  all.equal(n.A$normF, 0.52779033, tolerance=1e-9)
  all.equal(n.A.m, unname(as.matrix(n.A$mat)), tolerance = 1e-15)# seen rel.d.= 1.46e-16
})
set.seed(27)
m <- matrix(round(rnorm(25),2), 5, 5)
m <- m + t(m)
diag(m) <- pmax(0, diag(m)) + 1
(m <- round(cov2cor(m), 2))

str(near.m <- nearPD(m, trace = TRUE))
round(near.m$mat, 2)
norm(m - near.m$mat) # 1.102 / 1.08

if(requireNamespace("sfsmisc")) {
  m2 <- sfsmisc::posdefify(m) # a simpler approach
  norm(m - m2) # 1.185, i.e., slightly "less near"
}

round(nearPD(m, only.values=TRUE), 9)

## A longer example, extended from Jens' original,
## showing the effects of some of the options:

pr <- Matrix(c(1,      0.477, 0.644, 0.478, 0.651, 0.826,
               0.477, 1,      0.516, 0.233, 0.682, 0.75,
               0.644, 0.516, 1,      0.599, 0.581, 0.742,
               0.478, 0.233, 0.599, 1,      0.741, 0.8,
               0.651, 0.682, 0.581, 0.741, 1,      0.798,
               0.826, 0.75,  0.742, 0.8,   0.798, 1),
             nrow = 6, ncol = 6)

nc. <- nearPD(pr, conv.tol = 1e-7) # default
nc.$iterations # 2
nc.1 <- nearPD(pr, conv.tol = 1e-7, corr = TRUE)
nc.1$iterations # 11 / 12 (!)
```

```

ncr <- nearPD(pr, conv.tol = 1e-15)
str(ncr)# still 2 iterations
ncr.1 <- nearPD(pr, conv.tol = 1e-15, corr = TRUE)
ncr.1 $ iterations # 27 / 30 !

ncF <- nearPD(pr, conv.tol = 1e-15, conv.norm = "F")
stopifnot(all.equal(ncr, ncF))# norm type does not matter at all in this example

## But indeed, the 'corr = TRUE' constraint did ensure a better solution;
## cov2cor() does not just fix it up equivalently :
norm(pr - cov2cor(ncr$mat)) # = 0.09994
norm(pr -      ncr.1$mat)  # = 0.08746 / 0.08805

### 3) a real data example from a 'systemfit' model (3 eq.):
(load(system.file("external", "symW.rda", package="Matrix"))) # "symW"
dim(symW) # 24 x 24
class(symW)# "dsCMatrix": sparse symmetric
if(dev.interactive()) image(symW)
EV <- eigen(symW, only=TRUE)$values
summary(EV) ## looking more closely {EV sorted decreasingly}:
tail(EV)# all 6 are negative
EV2 <- eigen(sWpos <- nearPD(symW)$mat, only=TRUE)$values
stopifnot(EV2 > 0)
if(requireNamespace("sfsmisc")) {
  plot(pmax(1e-3,EV), EV2, type="o", log="xy", xaxt="n", yaxt="n")
  for(side in 1:2) sfsmisc::eaxis(side)
} else
  plot(pmax(1e-3,EV), EV2, type="o", log="xy")
abline(0, 1, col="red3", lty=2)

```

ngeMatrix-class

nMatrix

```

x "logical"
DimDimnames "integer"NULLMatrix
factors "list"

```

```

"ndenseMatrix""lMatrix""ndenseMatrix""denseMatrix""ndenseMatrix""Matrix"
"ndenseMatrix""Matrix""ndenseMatrix"

```

```

t()as(.)showMethods(class="ngeMatrix")

```

ntrMatrixnsyMatrixngCMatrix

```

showClass("ngeMatrix")
## "lgeMatrix" is really more relevant

```

nMatrix-class

nMatrix

Dim "integer"

Dimnames [character](#)Dim

```
signature(from = "matrix", to = "nMatrix")NATRUEsparseMatrix"nMatrix"  
nsparseMatrix
```

```
signature(e1 = "nMatrix", e2 = "....")
```

```
signature(e1 = "nMatrix", e2 = "....")
```

```
signature(e1 = "nMatrix", e2 = "....")
```

```
signature(e1 = "nMatrix", e2 = "....")
```

```
signature(x = "nMatrix", "....")
```

[lMatrix](#)[nsparseMatrix](#)[Matrix](#)

```
getClass("nMatrix")
```

```
L3 <- Matrix(upper.tri(diag(3)))
```

```
L3 # an "ltCMatrix"
```

```
as(L3, "nMatrix") # -> ntC*
```

```
## similar, not using Matrix()
```

```
as(upper.tri(diag(3)), "nMatrix")# currently "ngTMatrix"
```

nnzero-methods

[xMatrix](#)

```
nnzero(x, na.counted = NA)
```



```

x                Matrixnumeric
na.counted       logicalNA
na.counted       NA
na.counted       NANA
na.counted       NAX
na.counted       TRUE

```

```

xinteger

```

```

SsymmetricMatrixnnzero(S)length(S@x)

```

```

signature(x = "ANY") Matrix0xNANA.counted
signature(x = "denseMatrix") matrix"symmetricMatrix"
signature(x = "diagonalMatrix")signature(x = "indMatrix") "sparseMatrix"
signature(x = "sparseMatrix") "symmetricMatrix"

```

```

Matrixlengthlength(M)nnzero(M)M

```

```

drop0zapsmall

```

```

m <- Matrix(0+1:28, nrow = 4)
m[-3,c(2,4:5,7)] <- m[ 3, 1:4] <- m[1:3, 6] <- 0
(mT <- as(m, "TsparseMatrix"))
nnzero(mT)
(S <- crossprod(mT))
nnzero(S)
str(S) # slots are smaller than nnzero()
stopifnot(nnzero(S) == sum(as.matrix(S) != 0))# failed earlier

data(KNex, package = "Matrix")
M <- KNex$mm
class(M)
dim(M)
length(M); stopifnot(length(M) == prod(dim(M)))
nnzero(M) # more relevant than length
## the above are also visible from
str(M)

```

norm-methods

```
x"O""1""I""F""M""2"type
```

```
norm(x, type, ...)
```

```
x
```

```
type
```

```
  "O""o""1"  
  "I""i"  
  "F""f" x  
  "M""m" x  
  "2" svdx  
  "O"type[1]
```

```
...
```

```
dlangedlansydlantrzlangezlansyzlantr
```

```
"norm"type
```

```
onenormest()
```

```
norm()
```

```
x <- Hilbert(9)  
norm(x)# = "O" = "1"  
stopifnot(identical(norm(x), norm(x, "1")))  
norm(x, "I")# the same, because 'x' is symmetric
```

```
allnorms <- function(x) {  
  ## norm(NA, "2") did not work until R 4.0.0  
  do2 <- getRversion() >= "4.0.0" || !anyNA(x)  
  vapply(c("1", "I", "F", "M", if(do2) "2"), norm, 0, x = x)  
}  
allnorms(x)  
allnorms(Hilbert(10))
```

```
i <- c(1,3:8); j <- c(2,9,6:10); x <- 7 * (1:7)  
A <- sparseMatrix(i, j, x = x)          ## 8 x 10 "dgCMatrix"
```

```

(sA <- sparseMatrix(i, j, x = x, symmetric = TRUE)) ## 10 x 10 "dsCMatrix"
(tA <- sparseMatrix(i, j, x = x, triangular= TRUE)) ## 10 x 10 "dtCMatrix"
(allnorms(A) -> nA)
allnorms(sA)
allnorms(tA)
stopifnot(all.equal(nA, allnorms(as(A, "matrix"))),
  all.equal(nA, allnorms(tA))) # because tA == rbind(A, 0, 0)
A. <- A; A.[1,3] <- NA
stopifnot(is.na(allnorms(A.))) # gave error

```

nsparseMatrix-class

```
nsparseMatrixTRUEFALSETRUE
```

```

TsparseMatrixngTMatrixnsTMatrixntTMatrixCsparseMatrixngCMatrixnsCMatrix
ntCMatrixRsparseMatrixngRMatrixnsRMatrixntRMatrixgst

```

```
new("ngCMatrix", ...)
```

```

uplo "character"
diag "character""U""N"diag"U"
p "integer"
i "integer"
j "integer"
Dim "integer"

```

```

signature(from = "dgCMatrix", to = "ngCMatrix")"nsparseMatrix""nMatrix"TRUE
FALSE
signature(x = "ngCMatrix")x
signature(x = "lsparseMatrix")which(x, arr.ind)lMatrix

```

dgCMatrix

```

(m <- Matrix(c(0,0,2:0), 3,5, dimnames=list(LETTERS[1:3],NULL)))
## ``extract the nonzero-pattern of (m) into an nMatrix'':
nm <- as(m, "nsparseMatrix") ## -> will be a "ngCMatrix"
str(nm) # no 'x' slot
nnm <- !nm # no longer sparse
## consistency check:
stopifnot(xor(as( nm, "matrix"),
  as(nnm, "matrix")))

## low-level way of adding "non-structural zeros" :
nnm <- as(nnm, "lsparseMatrix") # "lgCMatrix"

```

```

nnm@x[2:4] <- c(FALSE, NA, NA)
nnm
as(nnm, "nMatrix") # NAs *and* non-structural 0 |---> 'TRUE'

data(KNex, package = "Matrix")
nnm <- as(KNex $ mm, "nMatrix")
str(xlx <- crossprod(nnm))# "nsCMatrix"
stopifnot(isSymmetric(xlx))
image(xlx, main=paste("crossprod(nnm) : Sparse", class(xlx)))

```

nsyMatrix-class

```
"nsyMatrix""nspMatrix"
```

```
new("nsyMatrix", ...)
```

```
uplo "character"
```

```
x "logical"
```

```
DimDimnames "integer"NULLMatrix
```

```
factors "list"
```

```
"nsyMatrix""ngeMatrix"
```

```
"nspMatrix""ndenseMatrix"
```

```
"symmetricMatrix""Matrix"showClass("nsyMatrix")
```

```
t()as(.)showMethods(class="nsyMatrix")
```

```
ngeMatrixMatrixt
```

```
(s0 <- new("nsyMatrix"))
```

```
(M2 <- Matrix(c(TRUE, NA, FALSE, FALSE), 2, 2)) # logical dense (ltr)
```

```
(sM <- M2 & t(M2)) # -> "lge"
```

```
class(sM <- as(sM, "nMatrix")) # -> "nge"
```

```
(sM <- as(sM, "symmetricMatrix")) # -> "nsy"
```

```
str(sM <- as(sM, "packedMatrix")) # -> "nsp", i.e., packed symmetric
```

ntrMatrix-class

```
"ntrMatrix""ntpMatrix"

x "logical"
uplo "character"
diag "character""U""N"triangularMatrix
DimDimnames "integer"NULLMatrix
factors "list"

"ntrMatrix""ngeMatrix"
"ntpMatrix""ndenseMatrix"
"triangularMatrix""denseMatrix""lMatrix"showClass("nsyMatrix")

t()as(.)showMethods(class="ntrMatrix")

ngeMatrixMatrix

showClass("ntrMatrix")

str(new("ntpMatrix"))
(nutr <- as(upper.tri(matrix(, 4, 4)), "ndenseMatrix"))
str(nutp <- pack(nutr)) # packed matrix: only 10 = 4*(4+1)/2 entries
!nutp # the logical negation (is not logical triangular !)
## but this one is:
stopifnot(all.equal(nutp, pack(!!nutp)))
```

pack-methods

```
pack()unpack()"packedMatrix""unpackedMatrix"

pack(x, ...)
## S4 method for signature 'dgeMatrix'
pack(x, symmetric = NA, upperTri = NA, ...)
## S4 method for signature 'lgeMatrix'
pack(x, symmetric = NA, upperTri = NA, ...)
## S4 method for signature 'ngeMatrix'
pack(x, symmetric = NA, upperTri = NA, ...)
## S4 method for signature 'matrix'
pack(x, symmetric = NA, upperTri = NA, ...)

unpack(x, ...)
```

```

x
      pack() "unpackedMatrix""matrix""packedMatrix"
      unpack() "packedMatrix""unpackedMatrix"
symmetric      NAx
upperTri      xNAx
...

pack(x)x"symmetricMatrix""triangularMatrix"isSymmetric()isTriangular()symmetric
upperTriTRUEFALSENA
pack()unpack()y <- unpack(pack(x))"unpackedMatrix"xidentical()

pack() "packedMatrix"x
unpack() "unpackedMatrix"x

showMethods("pack")
(s <- crossprod(matrix(sample(15), 5,3))) # traditional symmetric matrix
(sp <- pack(s))
mt <- as.matrix(tt <- tril(s))
(pt <- pack(mt))
stopifnot(identical(pt, pack(tt)),
  dim(s ) == dim(sp), all(s == sp),
  dim(mt) == dim(pt), all(mt == pt), all(mt == tt))

showMethods("unpack")
(cp4 <- chol(Hilbert(4))) # is triangular
tp4 <- pack(cp4) # [t]riangular [p]acked
str(tp4)
(unpack(tp4))
stopifnot(identical(tp4, pack(unpack(tp4))))

z1 <- new("dsyMatrix", Dim = c(2L, 2L), x = as.double(1:4), uplo = "U")
z2 <- unpack(pack(z1))
stopifnot(!identical(z1, z2), # _not_ identical
  all(z1 == z2)) # but mathematically equal
cbind(z1@x, z2@x) # (unused!) lower triangle is "lost" in translation

```

packedMatrix-class	"packedMatrix"
--------------------	----------------

```

"packedMatrix"choose(n+1,2) == n*(n+1)/2n"[dln]spMatrix""[dln]tpMatrix"
"dppMatrix"

```

```

uplo "character"
DimDimnames Matrix

```

```
"denseMatrix""Matrix""denseMatrix"
```

```
signature(x = "packedMatrix")
signature(x = "packedMatrix")
signature(object = "packedMatrix")
signature(object = "packedMatrix")
signature(object = "packedMatrix")
signature(x = "packedMatrix")
signature(x = "packedMatrix")
signature(x = "packedMatrix")
```

```
packunpack"unpackedMatrix""dspMatrix""ltpMatrix"
```

```
showClass("packedMatrix")
showMethods(classes = "packedMatrix")
```

pMatrix-class

```
pMatrixpMatrixindMatrix
```

```
permpMatrixmargin=1perminvPerm(perm)margin=2perminvPerm(perm)
nnPpermpM
```

PM	$P \%*\% M$	$M[p,]$
MP	$M \%*\% P$	$M[, i(p)]$
$P'M$	$\text{crossprod}(P, M)$	$M[i(p),]$
MP'	$\text{tcrossprod}(M, P)$	$M[, p]$
$P'P$	$\text{crossprod}(P)$	$\text{Diagonal}(n)$
PP'	$\text{tcrossprod}(P)$	$\text{Diagonal}(n)$

```
i := invPerm
```

```
new("pMatrix", ...)as(., "pMatrix")
```

```
marginperm indMatrixpermDim[1]1:Dim[1]
```

```
DimDimnames Matrix
```

```
"indMatrix"
```

```
%% signature(x = "pMatrix", y = "Matrix")showMethods("%*%", classes = "pMatrix")
```

```
coerce signature(from = "numeric", to = "pMatrix")pMatrix1:n
```

```
signature(x = "pMatrix")pMatrixpermmargin
```

```
signature(a = "pMatrix", b = "missing")pMatrixpermmarginshowMethods("solve",  
  classes = "pMatrix")
```

```
signature(x = "pMatrix", logarithm = "logical")
```

```
indMatrixinvPerm
```

```
(pm1 <- as(as.integer(c(2,3,1)), "pMatrix"))
```

```
t(pm1) # is the same as
```

```
solve(pm1)
```

```
pm1 %% t(pm1) # check that the transpose is the inverse
```

```
stopifnot(all(diag(3) == as(pm1 %% t(pm1), "matrix")),  
  is.logical(as(pm1, "matrix")))
```

```
set.seed(11)
```

```
## random permutation matrix :
```

```
(p10 <- as(sample(10), "pMatrix"))
```

```
## Permute rows / columns of a numeric matrix :
```

```
(mm <- round(array(rnorm(3 * 3), c(3, 3)), 2))
```

```
mm %% pm1
```

```
pm1 %% mm
```

```
try(as(as.integer(c(3,3,1)), "pMatrix"))# Error: not a permutation
```

```
as(pm1, "TsparseMatrix")
```

```
p10[1:7, 1:4] # gives an "ngTMatrix" (most economic!)
```

```
## row-indexing of a <pMatrix> keeps it as an <indMatrix>:
```

```
p10[1:3, ]
```

```
printSpMatrix
```

```
formatshowprintxprintSpMatrix2(x)printSpMatrix()x
```

```
printSpMatrix()formatSpMatrix()
```



```

formatSpMatrix(x, digits = NULL, maxp = 1e9,
               cld = getClassDef(class(x)), zero.print = ".",
               col.names, note.dropping.colnames = TRUE, uniDiag = TRUE,
               align = c("fancy", "right"), ...)

printSpMatrix(x, digits = NULL, maxp = max(100L, getOption("max.print")),
              cld = getClassDef(class(x)),
              zero.print = ".", col.names, note.dropping.colnames = TRUE,
              uniDiag = TRUE, col.trailer = "",
              align = c("fancy", "right"), ...)

printSpMatrix2(x, digits = NULL, maxp = max(100L, getOption("max.print")),
               zero.print = ".", col.names, note.dropping.colnames = TRUE,
               uniDiag = TRUE, supRows = NULL, supCols = NULL,
               col.trailer = if(supCols) "....." else "",
               align = c("fancy", "right"),
               width = getOption("width"), fitWidth = TRUE, ...)

x                sparseMatrix
digits           print.defaultNULLgetOption("digits")
maxp             options(max.print)
cld              xgetClassDef(class(x))
zero.print       ". "" ""0"print()
col.names        xoptions("sparse.colnames")FALSETRUE
                col.names"abb""sub""n"abb... <n>"abbreviate()substring()n
note.dropping.colnames
                col.namesFALSETRUE
uniDiag          "I""1"
col.trailer      show(<sparseMatrix>)
supRowssupCols   NULLprintSpMatrix2()NULLdim(x)options(c("width", "max.print"))
                FALSE
align           zero.print"fancy"zero.print = "."0align = "right"print(*, right
                = TRUE)
width           fitWidth
fitWidth        width
...

xmaxpx.formatSparseSimple()formatSparseMx

formatSpMatrix()
                col.names
printSpMatrix*()
                xinvisible

```

```
sparseMatrixsparseMatrixspMatrix
formatSparseM.formatSparseSimple()
```

```
f1 <- gl(5, 3, labels = LETTERS[1:5])
X <- as(f1, "sparseMatrix")
X ## <==> show(X) <==> print(X)
t(X) ## shows column names, since only 5 columns

X2 <- as(gl(12, 3, labels = paste(LETTERS[1:12], "c", sep=".")),
        "sparseMatrix")
X2
## less nice, but possible:
print(X2, col.names = TRUE) # use [,1] [,2] .. => does not fit

## Possibilities with column names printing:
t(X2) # suppressing column names
print(t(X2), col.names=TRUE)
print(t(X2), zero.print = "", col.names="abbr. 1")
print(t(X2), zero.print = "-", col.names="substring 2")
```

qr-methods

$$m \times n A \qquad P_1 A P_2 = Q R$$

$$A = P_1' Q R P_2'$$

$$P_1 P_2 Q = \prod_{j=1}^n H_j m \times m n H_j R m \times n$$

$$\text{denseMatrixqr.default} \text{dqr} \text{dcdgeqp3} P_1$$

$$\text{sparseMatrix} \text{cs_sqrcs_qrm} \geq n$$

```
qr(x, ...)
## S4 method for signature 'dgCMatrix'
qr(x, order = 3L, ...)
```

```
x          Matrixnrow(x) >= ncol(x)
order      0:3cs_sqrP_2A + A' \tilde{A}' \tilde{A} A' A \tilde{A} A A
...
```

$$xr < nx(n-r)n$$

$$P_1AP_2 = P_1 \begin{bmatrix} A_0 \\ 0 \end{bmatrix} P_2 = QR$$

$$A_0(m - (n - r)) \times n$$

QRqrqrxsparseMatrixsparseQR

sparseQR

dgCMatrix

qrqr.defaultqr

expand1expand2

CholeskyBunchKaufmanSchurlu

```
showMethods("qr", inherited = FALSE)
```

```
## Rank deficient: columns 3 {b2} and 6 {c3} are "extra"
```

```
M <- as(cbind(a1 = 1,
              b1 = rep(c(1, 0), each = 3L),
              b2 = rep(c(0, 1), each = 3L),
              c1 = rep(c(1, 0, 0), 2L),
              c2 = rep(c(0, 1, 0), 2L),
              c3 = rep(c(0, 0, 1), 2L)),
        "CsparseMatrix")
rownames(M) <- paste0("r", seq_len(nrow(M)))
b <- 1:6
eps <- .Machine$double.eps
```

```
## .... [1] full rank .....
## ==> a least squares solution of A x = b exists
##      and is unique _in exact arithmetic_
```

```
(A1 <- M[, -c(3L, 6L)])
(qr.A1 <- qr(A1))
```

```
stopifnot(exprs = {
  rankMatrix(A1) == ncol(A1)
  { d1 <- abs(diag(qr.A1@R)); sum(d1 < max(d1) * eps) == 0L }
  rcond(crossprod(A1)) >= eps
  all.equal(qr.coef(qr.A1, b), drop(solve(crossprod(A1), crossprod(A1, b))))
  all.equal(qr.fitted(qr.A1, b) + qr.resid(qr.A1, b), b)
})
```

```
## .... [2] numerically rank deficient with full structural rank .....
```

```

## ==> a least squares solution of  $Ax = b$  does not
##      exist or is not unique _in exact arithmetic_

(A2 <- M)
(qr.A2 <- qr(A2))

stopifnot(exprs = {
  rankMatrix(A2) == ncol(A2) - 2L
  { d2 <- abs(diag(qr.A2@R)); sum(d2 < max(d2) * eps) == 2L }
  rcond(crossprod(A2)) < eps

  ## 'qr.coef' computes unique least squares solution of "nearby" problem
  ##  $Zx = b$  for some full rank  $Z \sim A$ , currently without warning {FIXME} !
  tryCatch({ qr.coef(qr.A2, b); TRUE }, condition = function(x) FALSE)

  all.equal(qr.fitted(qr.A2, b) + qr.resid(qr.A2, b), b)
})

## .... [3] numerically and structurally rank deficient .....
## ==> factorization of _augmented_ matrix with
##      full structural rank proceeds as in [2]

## NB: implementation details are subject to change; see (*) below

A3 <- M
A3[, c(3L, 6L)] <- 0
A3
(qr.A3 <- qr(A3)) # with a warning ... "additional 2 row(s) of zeros"

stopifnot(exprs = {
  ## sparseQR object preserves the unaugmented dimensions (*)
  dim(qr.A3) == dim(A3)
  dim(qr.A3@V) == dim(A3) + c(2L, 0L)
  dim(qr.A3@R) == dim(A3) + c(2L, 0L)

  ## The augmented matrix remains numerically rank deficient
  rankMatrix(A3) == ncol(A3) - 2L
  { d3 <- abs(diag(qr.A3@R)); sum(d3 < max(d3) * eps) == 2L }
  rcond(crossprod(A3)) < eps
})

## Auxiliary functions accept and return a vector or matrix
## with dimensions corresponding to the unaugmented matrix (*),
## in all cases with a warning
qr.coef (qr.A3, b)
qr.fitted(qr.A3, b)
qr.resid (qr.A3, b)

## .... [4] yet more examples .....

## By disabling column pivoting, one gets the "vanilla" factorization
##  $A = Q^* R$ , where  $Q^* := P_1' Q$  is orthogonal because  $P_1$  and  $Q$  are

(qr.A1.pp <- qr(A1, order = 0L)) # partial pivoting

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

```

```

stopifnot(exprs = {
  length(qr.A1 @q) == ncol(A1)
  length(qr.A1.pp@q) == 0L # indicating no column pivoting
  ae2(A1[, qr.A1@q + 1L], qr.Q(qr.A1 ) %*% qr.R(qr.A1 ))
  ae2(A1      , qr.Q(qr.A1.pp) %*% qr.R(qr.A1.pp))
})

```

rankMatrix

$$n \times m \operatorname{Ark}(A) \operatorname{rk}(A) \leq \min(n, m)$$

```

rankMatrix(x, tol = NULL,
  method = c("tolNorm2", "qr.R", "qrLINPACK", "qr",
    "useGrad", "maybeGrad"),
  sval = svd(x, 0, 0)$d, warn.t = TRUE, warn.qr = TRUE)

```

```

qr2rankMatrix(qr, tol = NULL, isBqr = is.qr(qr), do.warn = TRUE)

```

x	$n \times m$
tol	methodmax(dim(x)) * .Machine\$double.eps method="tolNorm2"
method	<p>"tolNorm2" >= tol * max(sval)</p> <p>"qrLINPACK" qr(x, tol, LAPACK=FALSE)qr(...)\$rank</p> <p>x"qr.R"</p> <p>"qr.R" $Rqr()$qr-methods$QRd_iR d_i \geq \text{tol} \cdot \max(d_i)$</p> <p>"qr" x"qrLINPACK"x"qr.R"</p> <p>svalxsvalsvd()x</p> <p>"useGrad"</p> <p>"maybeGrad" "useGrad""tolNorm2"</p>
sval	xxmethod = "qr"
warn.t	rankMatrix() t (x)xmethod = "qr" t (x)
warn.qr	QR method"qr"rankMatrix()qr2rankMarix(.., do.warn = warn.qr)
qr	qr (x,...) class "qr"" sparseQR "
isBqr	logical qrqr() qr
do.warn	RQR

```

qr2rankMatrix()rankMatrix()"qr"methodqr

```

```

x01:min(dim(x))
 $QRd_i$ NaNwarn.qrdo.warnNANA_integer_

```

```

xsvlmethod = "qr"svalsvd()xdenseMatrix
xmethod = "qr"ditol = 0set.seed(42)

```

qrsvd

```

rankMatrix(cbind(1, 0, 1:3)) # 2

(meths <- eval(formals(rankMatrix)$method))

## a "border" case:
H12 <- Hilbert(12)
rankMatrix(H12, tol = 1e-20) # 12; but 11 with default method & tol.
sapply(meths, function(.m.) rankMatrix(H12, method = .m.))
## tolNorm2 qr.R qrLINPACK qr useGrad maybeGrad
##      11      11      12      12      11      11
## The meaning of 'tol' for method="qrLINPACK" and *dense* x is not entirely "scale free"
rMQL <- function(ex, M) rankMatrix(M, method="qrLINPACK",tol = 10^-ex)
rMQR <- function(ex, M) rankMatrix(M, method="qr.R",      tol = 10^-ex)
sapply(5:15, rMQL, M = H12) # result is platform dependent
## 7 7 8 10 10 11 11 11 12 12 12 {x86_64}
sapply(5:15, rMQL, M = 1000 * H12) # not identical unfortunately
## 7 7 8 10 11 11 12 12 12 12 12
sapply(5:15, rMQR, M = H12)
## 5 6 7 8 8 9 9 10 10 11 11
sapply(5:15, rMQR, M = 1000 * H12) # the *same*

## "sparse" case:
M15 <- kronecker(diag(x=c(100,1,10)), Hilbert(5))
sapply(meths, function(.m.) rankMatrix(M15, method = .m.))
#--> all 15, but 'useGrad' has 14.
sapply(meths, function(.m.) rankMatrix(M15, method = .m., tol = 1e-7)) # all 14

## "large" sparse
n <- 250000; p <- 33; nnz <- 10000
L <- sparseMatrix(i = sample.int(n, nnz, replace=TRUE),
                  j = sample.int(p, nnz, replace=TRUE),
                  x = rnorm(nnz))
(st1 <- system.time(r1 <- rankMatrix(L))) # warning+ ~1.5 sec (2013)
(st2 <- system.time(r2 <- rankMatrix(L, method = "qr"))) # considerably faster!
r1[[1]] == print(r2[[1]]) ## --> ( 33 TRUE )

## another sparse-"qr" one, which ``failed'' till 2013-11-23:
set.seed(42)
f1 <- factor(sample(50, 1000, replace=TRUE))
f2 <- factor(sample(50, 1000, replace=TRUE))
f3 <- factor(sample(50, 1000, replace=TRUE))
D <- t(do.call(rbind, lapply(list(f1,f2,f3), as, 'sparseMatrix')))
dim(D); nnzero(D) ## 1000 x 150 // 3000 non-zeros (= 2%)

```

```
stopifnot(rankMatrix(D, method='qr') == 148,
  rankMatrix(crossprod(D),method='qr') == 148)

## zero matrix has rank 0 :
stopifnot(sapply(meths, function(.m.)
  rankMatrix(matrix(0, 2, 2), method = .m.)) == 0)
```

rcond-methods

[showMethods\(rcond\)](#)

rcond(x, norm, ...)

```
## S4 method for signature 'sparseMatrix,character'
rcond(x, norm, useInv=FALSE, ...)
```

```
x          Matrix
norm       "0""0""1" useInv=TRUE normkind norm "I" norm
useInv     "Matrix" solve(x) 1/(||x|| · ||x-1||) x-1 x solve(x)
           solve(x)
           useInv
...

```

x

[norm](#)

A

$$\kappa(A) = \frac{\max_{\|v\|=1} \|Av\|}{\min_{\|v\|=1} \|Av\|}.$$

xrcond(qr.R(qr(X)), ...) Xxt(x)

rcond() 1/κ[0,1]

[normkappa\(\)](#) p = 2 [normsolve](#)
[condest](#)

```

x <- Matrix(rnorm(9), 3, 3)
rcond(x)
## typically "the same" (with more computational effort):
1 / (norm(x) * norm(solve(x)))
rcond(Hilbert(9)) # should be about 9.1e-13

## For non-square matrices:
rcond(x1 <- cbind(1,1:10))# 0.05278
rcond(x2 <- cbind(x1, 2:11))# practically 0, since x2 does not have full rank

## sparse
(S1 <- Matrix(rbind(0:1,0, diag(3:-2))))
rcond(S1)
m1 <- as(S1, "denseMatrix")
all.equal(rcond(S1), rcond(m1))

## wide and sparse
rcond(Matrix(cbind(0, diag(2:-1))))

## Large sparse example -----
m <- Matrix(c(3,0:2), 2,2)
M <- bdiag(kronecker(Diagonal(2), m), kronecker(m,m))
36*(iM <- solve(M)) # still sparse
MM <- kronecker(Diagonal(10), kronecker(Diagonal(5),kronecker(m,M)))
dim(M3 <- kronecker(bdiag(M,M),MM)) # 12'800 ^ 2
if(interactive()) ## takes about 2 seconds if you have >= 8 GB RAM
  system.time(r <- rcond(M3))
## whereas this is *fast* even though it computes solve(M3)
system.time(r. <- rcond(M3, useInv=TRUE))
if(interactive()) ## the values are not the same
  c(r, r.) # 0.05555 0.013888
## for all 4 norms available for sparseMatrix :
cbind(rr <- sapply(c("1","I","F","M"),
  function(N) rcond(M3, norm=N, useInv=TRUE)))

```

rep2abI

rep2abI(x, times)[rep.int](#)(x, times)[abIndex](#)

rep2abI(x, times)

x

times

[classabIndex](#)

`rep.int()``abIseq``abIndex`

```
(ab <- rep2abI(2:7, 4))
stopifnot(identical(as(ab, "numeric"),
  rep(2:7, 4)))
```

`rleDiff`-class

`"rleDiff"``x``diff(x)``rle()`

`new("rleDiff", ...)`

`first` `"numLike"``"numeric"``"logical"`

`rle` `"rle"``list``"lengths"``"values"``rle()``lengths``double`

`show`

`abIndex`

`rleabIndex`

`showClass("rleDiff")`

```
ab <- c(abIseq(2, 100), abIseq(20, -2))
ab@rleD # is "rleDiff"
```

`rsparsematrix`

```
rand.x = NULLnsparseMatrix
```

```
rsparsematrix(nrow, ncol, density, nnz = round(density * maxE),
              symmetric = FALSE,
              rand.x = function(n) signif(rnorm(n), 2), ...)
```

```
nrowncol      dim
density       [0,1]nnznnz
nnz           nrow*ncoldensity
symmetric     symmetricMatrixnnz
rand.x        NULLxfunctionrand.x(n)rand.x = rnormrand.x = runif
...           sparseMatrix()repr
```

```
(i,j)symmetricssample.int(nrow*ncol,      nnz)rand.xNULLx      <-      rand.x(nnz)
sparseMatrix(i=i,  j=j,  x=x,  ..)rand.x=NULLsparseMatrix(i=i,  j=j,  ..)
nsparseMatrix
```

```
sparseMatrixMdim(M) == c(nrow, ncol)symmetriccnzM <- nnzero(M)nzM <= nnznzM ==
nnz
```

```
set.seed(17)# to be reproducible
M <- rsparsematrix(8, 12, nnz = 30) # small example, not very sparse
M
M1 <- rsparsematrix(1000, 20, nnz = 123, rand.x = runif)
summary(M1)

## a random *symmetric* Matrix
(S9 <- rsparsematrix(9, 9, nnz = 10, symmetric=TRUE)) # dsCMatrix
nnzero(S9)# ~ 20: as 'nnz' only counts one "triangle"

## a random patter*n* aka boolean Matrix (no 'x' slot):
(n7 <- rsparsematrix(5, 12, nnz = 10, rand.x = NULL))

## a [T]riplet representation sparseMatrix:
T2 <- rsparsematrix(40, 12, nnz = 99, repr = "T")
head(T2)
```

RsparseMatrix-class

```
"RsparseMatrix"showClass("RsparseMatrix")
```

```
j "integer"nnzero  
p "integer"  
DimDimnames sparseMatrix
```

```
"sparseMatrix""Matrix""sparseMatrix"
```

```
CsparseMatrix"RsparseMatrix"R[i, j] <- v"RsparseMatrix"RR"TsparseMatrix"  
signature(x = "RsparseMatrix")  
signature(from = "RsparseMatrix", to = "CsparseMatrix")  
signature(from = "RsparseMatrix", to = "TsparseMatrix")
```

```
sparseMatrixxdgRMatrix
```

```
showClass("RsparseMatrix")
```

Schur-class

Schurn $\times nA$

$$A = QTQ'$$

$QT1 \times 12 \times 2AQATA$

A

ATT

```
DimDimnames MatrixFactorization  
Q Matrix  
T Matrix  
EValues TT
```

SchurFactorizationMatrixFactorizationSchurFactorization

new("Schur", ...)Schur(x)xMatrixdgeMatrix

determinant signature(from = "Schur", logarithm = "logical")A

expand1 signature(x = "Schur")expand1-methods

expand2 signature(x = "Schur")expand2-methods

solve signature(a = "Schur", b = .)solve-methods

<https://netlib.org/lapack/double/dgees.f>

dgeMatrix

Schurexpand1expand2

```
showClass("Schur")
set.seed(0)

n <- 4L
(A <- Matrix(rnorm(n * n), n, n))

## With dimnames, to see that they are propagated :
dimnames(A) <- list(paste0("r", seq_len(n)),
                    paste0("c", seq_len(n)))

(sch.A <- Schur(A))
str(e.sch.A <- expand2(sch.A), max.level = 2L)

## A ~ Q T Q' in floating point
stopifnot(exprs = {
  identical(names(e.sch.A), c("Q", "T", "Q."))
  all.equal(A, with(e.sch.A, Q %*% T %*% Q.))
})

## Factorization handled as factorized matrix
b <- rnorm(n)
stopifnot(all.equal(det(A), det(sch.A)),
          all.equal(solve(A, b), solve(sch.A, b)))

## One of the non-general cases:
Schur(Diagonal(6L))
```

Schur-methods

$n \times n$ A

$$A = QTQ'$$

$QT1 \times 12 \times 2AQATA$

dgees

```
Schur(x, vectors = TRUE, ...)
```

x **Matrix**

vectors TRUE

...

SchurFactorizationvectors = TRUE**Schur**xQTE**Values****Schur**

vectors = FALSEQ

<https://netlib.org/lapack/double/dgees.f>

Schur

dgeMatrix

expand1expand2

CholeskyBunchKaufmanluqr

```
showMethods("Schur", inherited = FALSE)
set.seed(0)
```

```
Schur(Hilbert(9L)) # real eigenvalues
```

```
(A <- Matrix(round(rnorm(25L, sd = 100)), 5L, 5L))
(sch.A <- Schur(A)) # complex eigenvalues
```

```
## A ~ Q T Q' in floating point
str(e.sch.A <- expand2(sch.A), max.level = 2L)
stopifnot(all.equal(A, Reduce("%*%", e.sch.A)))
```

```
(e1 <- eigen(sch.A@T, only.values = TRUE)$values)
(e2 <- eigen(    A   , only.values = TRUE)$values)
(e3 <- sch.A@EValues)
```

```

stopifnot(exprs = {
  all.equal(e1, e2, tolerance = 1e-13)
  all.equal(e1, e3[order(Mod(e3), decreasing = TRUE)], tolerance = 1e-13)
  identical(Schur(A, vectors = FALSE),
    list(T = sch.A@T, EValues = e3))
  identical(Schur(as(A, "matrix")),
    list(Q = as(sch.A@Q, "matrix"),
      T = as(sch.A@T, "matrix"), EValues = e3))
})

```

solve-methods

solve

[solve](#) $XAX = BAXBA$

`solve(a, b, ...)`

S4 method for signature 'dgeMatrix,ANY'
`solve(a, b, tol = .Machine$double.eps, ...)`

S4 method for signature 'dgCMatrix,missing'
`solve(a, b, sparse = TRUE, ...)`
 ## S4 method for signature 'dgCMatrix,matrix'
`solve(a, b, sparse = FALSE, ...)`
 ## S4 method for signature 'dgCMatrix,denseMatrix'
`solve(a, b, sparse = FALSE, ...)`
 ## S4 method for signature 'dgCMatrix,sparseMatrix'
`solve(a, b, sparse = TRUE, ...)`

S4 method for signature 'denseLU,dgeMatrix'
`solve(a, b, ...)`
 ## S4 method for signature 'BunchKaufman,dgeMatrix'
`solve(a, b, ...)`
 ## S4 method for signature 'Cholesky,dgeMatrix'
`solve(a, b, ...)`
 ## S4 method for signature 'sparseLU,dgCMatrix'
`solve(a, b, tol = .Machine$double.eps, ...)`
 ## S4 method for signature 'sparseQR,dgCMatrix'
`solve(a, b, ...)`
 ## S4 method for signature 'CHMfactor,dgCMatrix'
`solve(a, b, system = c("A", "LDLt", "LD", "DLt", "L", "Lt", "D", "P", "Pt"), ...)`

<code>a</code>	MatrixMatrixFactorization	
<code>b</code>	sparseVectorMatrix <code>NROW(b) == nrow(a)</code> <code>length(b) ==</code>	
<code>tol</code>	adenseMatrix <code>rcond</code> <code>atol</code> <code>asparseLU</code> <code>min(d)/max(d)</code> <code>told</code>	<code>=</code>
	<code>abs(diag(a@U))</code> <code>tol = 0</code>	
<code>sparse</code>	sparseMatrix <code>abbsparse = TRUE</code> <code>bsparse = FALSE</code>	
<code>system</code>	aCHMfactor	
<code>...</code>		

```

aaaa
aabab
ad..Matrix..=(ge|tr|tp|sy|sp|po|pp)d..trid..trsb
acs_lsolvecs_usolvecs_spsolvecholmod_solvecholmod_spsolvebbbsparseabb
asparseQRqr.coef
aCHMfactor  $AX = B$  systemCHMfactor-class

```

system	isLDL(a)=TRUE	isLDL(a)=FALSE
"A"	$AX = B$	$AX = B$
"LDLt"	$L_1 D L_1' X = B$	$LL' X = B$
"LD"	$L_1 D X = B$	$LX = B$
"DLt"	$D L_1' X = B$	$L' X = B$
"L"	$L_1 X = B$	$LX = B$
"Lt"	$L_1' X = B$	$L' X = B$
"D"	$DX = B$	$X = B$
"P"	$X = P_1 B$	$X = P_1 B$
"Pt"	$X = P_1' B$	$X = P_1' B$

```

MatrixFactorization
CholeskyBunchKaufmanSchurluqr
solve
qr.coef

```

```

## A close to symmetric example with "quite sparse" inverse:
n1 <- 7; n2 <- 3
dd <- data.frame(a = gl(n1,n2), b = gl(n2,1,n1*n2))# balanced 2-way
X <- sparse.model.matrix(~ -1+ a + b, dd)# no intercept --> even sparser
XXt <- tcrossprod(X)
diag(XXt) <- rep(c(0,0,1,0), length.out = nrow(XXt))

n <- nrow(ZZ <- kronecker(XXt, Diagonal(x=c(4,1))))
image(a <- 2*Diagonal(n) + ZZ %*% Diagonal(x=c(10, rep(1, n-1))))
isSymmetric(a) # FALSE
image(drop0(skewpart(a)))
image(ia0 <- solve(a, tol = 0)) # checker board, dense [but really, a is singular!]
try(solve(a, sparse=TRUE))##-> error [ TODO: assertError ]
ia. <- solve(a, sparse=TRUE, tol = 1e-19)##-> *no* error
if(R.version$arch == "x86_64")
  ## Fails on 32-bit [Fedora 19, R 3.0.2] from Matrix 1.1-0 on [FIXME ??] only
  stopifnot(all.equal(as.matrix(ia.), as.matrix(ia0)))
a <- a + Diagonal(n)
iad <- solve(a)
ias <- solve(a, sparse=FALSE)
stopifnot(all.equal(as(iad,"denseMatrix"), ias, tolerance=1e-14))
I. <- iad %*% a ; image(I.)
I0 <- drop0(zapsmall(I.)); image(I0)
.I <- a %*% iad

```

```
.I0 <- drop0(zapsmall(.I))
stopifnot( all.equal(as(I0, "diagonalMatrix"), Diagonal(n)),
           all.equal(as(.I0,"diagonalMatrix"), Diagonal(n)) )
```

sparse.model.matrix

```
sparse.model.matrixfac2sparse
```

```
fac2[Ss]parse()sparse.model.matrix()
```

```
sparse.model.matrix(object, data = environment(object),
                     contrasts.arg = NULL, xlev = NULL, transpose = FALSE,
                     drop.unused.levels = FALSE, row.names = TRUE,
                     sep = "", verbose = FALSE, ...)
```

```
fac2sparse(from, to = c("d", "l", "n"),
            drop.unused.levels = TRUE, repr = c("C", "R", "T"), giveCsparse)
fac2Sparse(from, to = c("d", "l", "n"),
            drop.unused.levels = TRUE, repr = c("C", "R", "T"), giveCsparse,
            factorPatt12, contrasts.arg = NULL)
```

object

data [model.frame](#)[model.frame](#)

contrasts.arg [sparse.model.matrix\(\)](#) [contrasts](#)[data](#)[factor](#)
[fac2Sparse\(\)](#) [NULL](#) ["sparseMatrix"](#)

xlev [model.frame](#)[data](#) ["terms"](#)

transpose [transpose = TRUE](#)

drop.unused.levels
[sparse.model.matrix](#)[FALSE](#)[model.matrix\(\)](#)

row.names

sep [character](#)[paste\(\)](#)

verbose

...

from [fac2sparse\(\)](#)[factor](#)

to ["d"](#)[double](#)

giveCsparse [repr](#)[CsparseMatrix](#)

repr [character](#) ["C"](#) ["T"](#) ["R"](#) [CsparseMatrix](#) [TsparseMatrix](#) [RsparseMatrix](#)

factorPatt12 [fpfp\[1\]](#)[t\(X\)](#)[fp\[2\]](#)[t\(X\)](#) [fac2sparse\(\)](#)

[CsparseMatrix](#)[fac2sparse\(\)](#)[repr = "C"](#) [TsparseMatrix](#) [RsparseMatrix](#)

[fac2Sparse\(\)](#)[list](#)[factorPatt12](#)

[fac2sparse\(\)](#)[sparse.model.matrix\(\)](#)[t](#)


```
model.Matrix(sparse = TRUE)sparse.model.matrixmodel.MatrixmodelMatrixassign
contrasts
```

`model.matrix`

`model.Matrix`

```
as(f, "sparseMatrix")coerce(from = "factor", ..)f
```

```
dd <- data.frame(a = gl(3,4), b = gl(4,1,12))# balanced 2-way
options("contrasts") # the default: "contr.treatment"
sparse.model.matrix(~ a + b, dd)
sparse.model.matrix(~ -1+ a + b, dd)# no intercept --> even sparser
sparse.model.matrix(~ a + b, dd, contrasts = list(a="contr.sum"))
sparse.model.matrix(~ a + b, dd, contrasts = list(b="contr.SAS"))
```

```
## Sparse method is equivalent to the traditional one :
stopifnot(all(sparse.model.matrix(~ a + b, dd) ==
              Matrix(model.matrix(~ a + b, dd), sparse=TRUE)),
          all(sparse.model.matrix(~0 + a + b, dd) ==
              Matrix(model.matrix(~0 + a + b, dd), sparse=TRUE)))
```

```
(ff <- gl(3,4,, c("X","Y", "Z")))
fac2sparse(ff) # 3 x 12 sparse Matrix of class "dgCMatrix"
##
## X 1 1 1 1 . . . . .
## Y . . . . 1 1 1 1 . . . .
## Z . . . . . 1 1 1 1
```

```
## can also be computed via sparse.model.matrix():
f30 <- gl(3,0 )
f12 <- gl(3,0, 12)
stopifnot(
  all.equal(t( fac2sparse(ff) ),
            sparse.model.matrix(~ 0+ff),
            tolerance = 0, check.attributes=FALSE),
  is(M <- fac2sparse(f30, drop= TRUE),"CsparseMatrix"), dim(M) == c(0, 0),
  is(M <- fac2sparse(f30, drop=FALSE),"CsparseMatrix"), dim(M) == c(3, 0),
  is(M <- fac2sparse(f12, drop= TRUE),"CsparseMatrix"), dim(M) == c(0,12),
  is(M <- fac2sparse(f12, drop=FALSE),"CsparseMatrix"), dim(M) == c(3,12)
)
```

`sparseLU` $n \times n A$

$$P_1 A P_2 = LU$$

$$A = P_1' L U P_2'$$

$P_1 P_2 L U$

DimDimnames [MatrixFactorization](#)

L [dtCMatrix](#) L

U [dtCMatrix](#) U

pq Dim[1]q $P_1 A P_2 A[p+1, q+1] A[p+1,] q$

[LUMatrixFactorizationLU](#)

`new("sparseLU", ...)` [lu\(x\)](#) [xsparseMatrix](#) [dgCMatrix](#)

determinant signature(from = "sparseLU", logarithm = "logical") A

expand signature(x = "sparseLU") [expand-methods](#)

expand1 signature(x = "sparseLU") [expand1-methods](#)

expand2 signature(x = "sparseLU") [expand2-methods](#)

solve signature(a = "sparseLU", b = .) [solve-methods](#)

[denseLU](#)

[dgCMatrix](#)

[luexpand1expand2](#)

`showClass("sparseLU")`

`set.seed(2)`

```
A <- as(readMM(system.file("external", "pores_1.mtx", package = "Matrix")),
        "CsparseMatrix")
```

```
(n <- A@Dim[1L])
```

```
## With dimnames, to see that they are propagated :
dimnames(A) <- dn <- list(paste0("r", seq_len(n)),
                          paste0("c", seq_len(n)))
```

```

(lu.A <- lu(A))
str(e.lu.A <- expand2(lu.A), max.level = 2L)

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

## A ~ P1' L U P2' in floating point
stopifnot(exprs = {
  identical(names(e.lu.A), c("P1.", "L", "U", "P2."))
  identical(e.lu.A[["P1."]],
    new("pMatrix", Dim = c(n, n), Dimnames = c(dn[1L], list(NULL)),
      margin = 1L, perm = invertPerm(lu.A@p, 0L, 1L)))
  identical(e.lu.A[["P2."]],
    new("pMatrix", Dim = c(n, n), Dimnames = c(list(NULL), dn[2L]),
      margin = 2L, perm = invertPerm(lu.A@q, 0L, 1L)))
  identical(e.lu.A[["L"]], lu.A@L)
  identical(e.lu.A[["U"]], lu.A@U)
  ae1(A, with(e.lu.A, P1. %*% L %*% U %*% P2.))
  ae2(A[lu.A@p + 1L, lu.A@q + 1L], with(e.lu.A, L %*% U))
})

## Factorization handled as factorized matrix
b <- rnorm(n)
stopifnot(identical(det(A), det(lu.A)),
  identical(solve(A, b), solve(lu.A, b)))

```

sparseMatrix

[classCsparseMatrix](#)[RsparseMatrix](#)[TsparseMatrix](#)

[new\("..\[CRT\]Matrix", ...\)](#)

```

sparseMatrix(i, j, p, x, dims, dimnames,
  symmetric = FALSE, triangular = FALSE, index1 = TRUE,
  repr = c("C", "R", "T"), giveCsparse,
  check = TRUE, use.last.ij = FALSE)

```

<code>ij</code>	$\text{TRUE} x_k(i_k, j_k)$ TsparseMatrix <code>use.last.ij</code> $\text{TRUE} x_k$
<code>p</code>	<code>ijp</code>
<code>x</code>	<code>ijx</code> nsparseMatrix
<code>dims</code>	<code>!index1 + c(max(i), max(j))</code>
<code>dimnames</code>	dimnames <code>NULL</code>
<code>symmetric</code>	(i, j, p)
<code>triangular</code>	(i, j, p)
<code>index1</code>	<code>TRUE</code> <code>ij</code> <code>FALSE</code>
<code>repr</code>	character <code>"C"</code> <code>"R"</code> <code>"T"</code> CsparseMatrix RsparseMatrix TsparseMatrix
<code>giveCsparse</code>	<code>repr</code> CsparseMatrix TsparseMatrix CsparseMatrix
<code>check</code>	<code>FALSE</code>
<code>use.last.ij</code>	(i_k, j_k) <code>FALSE</code> TsparseMatrix

```

ijp
pijx
ijpprep(seq_along(dp),dp)dp <- diff(p)
singulartriangularDiagonal()
ijpindexlijTsparseMatrixx(i_k,j_k)x_kTsparseMatrixuse.last.ij
repr = "C"CsparseMatrixrepr = "R"RsparseMatrixrepr = "T"TsparseMatrix
CsparseMatrixxCsparseMatrix

```

CsparseMatrixgeneralMatrix

```

index1 = FALSE+ 1ijij

```

```

Matrix(*, sparse=TRUE)sparseMatrix()bdiagDiagonalbandSparse
rsparsematrix()
xtabs(*, sparse=TRUE)sparse.model.matrix()
CsparseMatrix

```

```

## simple example
i <- c(1,3:8); j <- c(2,9,6:10); x <- 7 * (1:7)
(A <- sparseMatrix(i, j, x = x)) ## 8 x 10 "dgCMatrix"
summary(A)
str(A) # note that *internally* 0-based row indices are used

(sA <- sparseMatrix(i, j, x = x, symmetric = TRUE)) ## 10 x 10 "dsCMatrix"
(tA <- sparseMatrix(i, j, x = x, triangular= TRUE)) ## 10 x 10 "dtCMatrix"
stopifnot( all(sA == tA + t(tA)) ,
           identical(sA, as(tA + t(tA), "symmetricMatrix")))

## dims can be larger than the maximum row or column indices
(AA <- sparseMatrix(c(1,3:8), c(2,9,6:10), x = 7 * (1:7), dims = c(10,20)))
summary(AA)

## i, j and x can be in an arbitrary order, as long as they are consistent
set.seed(1); (perm <- sample(1:7))
(A1 <- sparseMatrix(i[perm], j[perm], x = x[perm]))
stopifnot(identical(A, A1))

## The slots are 0-index based, so
try( sparseMatrix(i=A@i, p=A@p, x= seq_along(A@x)) )
## fails and you should say so: 1-indexing is FALSE:
sparseMatrix(i=A@i, p=A@p, x= seq_along(A@x), index1 = FALSE)

## the (i,j) pairs can be repeated, in which case the x's are summed
(args <- data.frame(i = c(i, 1), j = c(j, 2), x = c(x, 2)))
(Aa <- do.call(sparseMatrix, args))

```

```

## explicitly ask for elimination of such duplicates, so
## that the last one is used:
(A. <- do.call(sparseMatrix, c(args, list(use.last.ij = TRUE))))
stopifnot(Aa[1,2] == 9, # 2+7 == 9
          A.[1,2] == 2) # 2 was *after* 7

## for a pattern matrix, of course there is no "summing":
(nA <- do.call(sparseMatrix, args[c("i","j")]))

dn <- list(LETTERS[1:3], letters[1:5])
## pointer vectors can be used, and the (i,x) slots are sorted if necessary:
m <- sparseMatrix(i = c(3,1, 3:2, 2:1), p = c(0:2, 4,4,6), x = 1:6, dimnames = dn)
m
str(m)
stopifnot(identical(dimnames(m), dn))

sparseMatrix(x = 2.72, i=1:3, j=2:4) # recycling x
sparseMatrix(x = TRUE, i=1:3, j=2:4) # recycling x, |--> "lgCMatrix"

## no 'x' --> patter*n* matrix:
(n <- sparseMatrix(i=1:6, j=rev(2:7)))# -> ngCMatrix

## an empty sparse matrix:
(e <- sparseMatrix(dims = c(4,6), i={}, j={}))

## a symmetric one:
(sy <- sparseMatrix(i= c(2,4,3:5), j= c(4,7:5,5), x = 1:5,
                    dims = c(7,7), symmetric=TRUE))
stopifnot(isSymmetric(sy),
          identical(sy, ## switch i <-> j {and transpose }
                    t( sparseMatrix(j= c(2,4,3:5), i= c(4,7:5,5), x = 1:5,
                                   dims = c(7,7), symmetric=TRUE))))

## rsparsematrix() calls sparseMatrix() :
M1 <- rsparsematrix(1000, 20, nnz = 200)
summary(M1)

## pointers example in converting from other sparse matrix representations.
if(requireNamespace("SparseM") &&
   packageVersion("SparseM") >= "0.87" &&
   nzchar(dfil <- system.file("extdata", "rua_32_ax.rua", package = "SparseM"))) {
  X <- SparseM::model.matrix(SparseM::read.matrix.hb(dfil))
  XX <- sparseMatrix(j = X@ja, p = X@ia - 1L, x = X@ra, dims = X@dimension)
  validObject(XX)

  ## Alternatively, and even more user friendly :
  X. <- as(X, "Matrix") # or also
  X2 <- as(X, "sparseMatrix")
  stopifnot(identical(XX, X.), identical(X., X2))
}

```

```
Dim "integer"
Dimnames MatrixMatrix
```

```
"Matrix"
```

```
(object = "sparseMatrix")show".printSpMatrix()
signature(x = "sparseMatrix")
  printshow()printSpMatrix()
signature(x = "sparseMatrix")
  formatformatSpMatrix()
(object = "sparseMatrix", uniqT=FALSE)"sparseSummary"data.frame(i,j,x)(i,j)
  nsparseMatrixprintwriteMM
(x = *, y = *)cbindrbind

(x = *, y = *)cbind2
(x = "sparseMatrix", logarithm=TRUE)determinant()Choleskylu
(x = "sparseMatrix")
signature(x = "sparseMatrix", value = "ANY")valueprod(value) == prod(dim(x))
signature(from = "factor", to = "sparseMatrix")"sparseMatrix""dgCMatrix""x"NA0
colSumsnorm
```

```
%*%crossproddgeMatrix
```

```
sparseMatrixxtabs(*, sparse=TRUE)sparse.model.matrix()
T2graph"graph"
```

```
showClass("sparseMatrix") ## and look at the help() of its subclasses
M <- Matrix(0, 10000, 100)
M[1,1] <- M[2,3] <- 3.14
M ## show(.) method suppresses printing of the majority of rows
```

```
data(CAex, package = "Matrix")
dim(CAex) # 72 x 72 matrix
determinant(CAex) # works via sparse lu(.)
```

```
## factor -> t( <sparse design matrix> ) :
(fact <- gl(5, 3, 30, labels = LETTERS[1:5]))
(Xt <- as(fact, "sparseMatrix")) # indicator rows
```

```
## missing values --> all-0 columns:
f.mis <- fact
i.mis <- c(3:5, 17)
is.na(f.mis) <- i.mis
Xt != (X. <- as(f.mis, "sparseMatrix")) # differ only in columns 3:5,17
stopifnot(all(X[,i.mis] == 0), all(Xt[,-i.mis] == X[,-i.mis]))
```

sparseQR-class

$\text{sparseQR} m \times nm \geq n$

$$P_1 A P_2 = Q R = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1$$

$$A = P_1' Q R P_2' = P_1' \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} P_2' = P_1' Q_1 R_1 P_2'$$

$$P_1 P_2 Q = \prod_{j=1}^n H_j m \times m Q_1 n n H_j R m \times n R_1 n$$

`qrR(qr, complete = FALSE, backPermute = TRUE, row.names = TRUE)`

<code>qr</code>	<code>sparseQR</code> <code>qrx</code>
<code>complete</code>	<code>RR₁</code>
<code>backPermute</code>	<code>RR₁P'₂</code>
<code>row.names</code>	<code>dimnames(qr)[1]complete = FALSE</code> <code>n</code>

`qr.QQ``P'1Q``qr.default``P1qr.Q(qr.default(x))``P'1Q`
`qr.qyqr.qty``P'1QQ'``P1QQ'`
`qr.Qqr.Rqr.qyqr.qtyqr(x)qr(as(x, "matrix"))``xdgCMatrix``P1P2`
`qr.Xqr.coefqr.fittedqr.resid`
`qr.RqrRbackPermute = TRUE``1.6-0``qrR>= 1.6-0`

`Dim``Dimnames` `MatrixFactorization`
`beta` `Dim[2]``V`
`V` `dgCMatrix``Dim[2]``nrow(V)``Dim[1]``Dim[1]+Dim[2]``VHjQHjdiag(Dim[1]) - beta[j] *`
`tcrossprod(V[, j])`
`R` `dgCMatrix``nrow(V)``Dim[2]``RR`
`pq` `nrow(V)``Dim[2]``q``P1AP2A[p+1, q+1]``A[p+1,]q`

`QRMatrixFactorizationQR`

`new("sparseQR", ...)``qr(x)``x``sparseMatrix``dgCMatrix`

```

determinant signature(from = "sparseQR", logarithm = "logical")A
expand1 signature(x = "sparseQR")expand1-methods
expand2 signature(x = "sparseQR")expand2-methods
qr.Q signature(qr = "sparseQR")dgeMatrix $P_1'QP_1'Q_1$ completeFALSE $P_1'Q_1$ 
qr.R signature(qr = "sparseQR") $qrRRR_1RP_2'R_1P_2'$ completebackPermuteFALSE $R_1$ 
dtCMatrixxdgCMatrix
qr.X signature(qr = "sparseQR") $A$ dgeMatrix $m > n$  $ncolnP_1'QJJ(n+1)ncolm \times m$ 
qr.coef signature(qr = "sparseQR", y = .)dgeMatrix $yP_2R_1^{-1}Q_1'P_1$ 
qr.fitted signature(qr = "sparseQR", y = .)dgeMatrix $yP_1'Q_1Q_1'P_1$ 
qr.resid signature(qr = "sparseQR", y = .)dgeMatrix $yP_1'Q_2Q_2'P_1$ 
qr.qty signature(qr = "sparseQR", y = .)dgeMatrix $yQ'P_1$ 
qr.qy signature(qr = "sparseQR", y = .)dgeMatrix $yP_1'Q$ 
solve signature(a = "sparseQR", b = .)solve-methods

```

[dgCMatrix](#)

[qrqr.defaultqr](#)

[qr-methods](#)

[expand1](#)[expand2](#)

[qr.Qqr.Rqr.Xqr.coefqr.fittedqr.residqr.qtyqr.qyqr.solve](#)

```
showClass("sparseQR")
```

```
set.seed(2)
```

```
m <- 300L
```

```
n <- 60L
```

```
A <- rsparsematrix(m, n, 0.05)
```

```
## With dimnames, to see that they are propagated :
```

```
dimnames(A) <- dn <- list(paste0("r", seq_len(m)),
                          paste0("c", seq_len(n)))
```

```
(qr.A <- qr(A))
```

```
str(e.qr.A <- expand2(qr.A, complete = FALSE), max.level = 2L)
```

```
str(E.qr.A <- expand2(qr.A, complete = TRUE), max.level = 2L)
```

```
t(sapply(e.qr.A, dim))
```

```
t(sapply(E.qr.A, dim))
```

```
## Horribly inefficient, but instructive :
```

```
slowQ <- function(V, beta) {
  d <- dim(V)
```



```

Q <- diag(d[1L])
if(d[2L] > 0L) {
  for(j in d[2L]:1L) {
    cat(j, "\n", sep = "")
    Q <- Q - (beta[j] * tcrossprod(V[, j])) %*% Q
  }
}
Q

}

ae1 <- function(a, b, ...) all.equal(as(a, "matrix"), as(b, "matrix"), ...)
ae2 <- function(a, b, ...) ae1(unname(a), unname(b), ...)

## A ~ P1' Q R P2' ~ P1' Q1 R1 P2' in floating point
stopifnot(exprs = {
  identical(names(e.qr.A), c("P1.", "Q1", "R1", "P2."))
  identical(names(E.qr.A), c("P1.", "Q", "R", "P2."))
  identical(e.qr.A[["P1."]],
    new("pMatrix", Dim = c(m, m), Dimnames = c(dn[1L], list(NULL)),
      margin = 1L, perm = invertPerm(qr.A@p, 0L, 1L)))
  identical(e.qr.A[["P2."]],
    new("pMatrix", Dim = c(n, n), Dimnames = c(list(NULL), dn[2L]),
      margin = 2L, perm = invertPerm(qr.A@q, 0L, 1L)))
  identical(e.qr.A[["R1"]], triu(E.qr.A[["R"]][seq_len(n), ]))
  identical(e.qr.A[["Q1"]], E.qr.A[["Q"]][seq_len(n), ])
  identical(E.qr.A[["R"]], qr.A@R)
  ## ae1(E.qr.A[["Q"]], slowQ(qr.A@V, qr.A@beta))
  ae1(crossprod(E.qr.A[["Q"]]), diag(m))
  ae1(A, with(e.qr.A, P1. %*% Q1 %*% R1 %*% P2.))
  ae1(A, with(E.qr.A, P1. %*% Q %*% R %*% P2.))
  ae2(A.perm <- A[qr.A@p + 1L, qr.A@q + 1L], with(e.qr.A, Q1 %*% R1))
  ae2(A.perm, with(E.qr.A, Q %*% R ))
})

## More identities
b <- rnorm(m)
stopifnot(exprs = {
  ae1(qrX <- qr.X(qr.A), A)
  ae2(qrQ <- qr.Q(qr.A), with(e.qr.A, P1. %*% Q1))
  ae2(qrR <- qr.R(qr.A), with(e.qr.A, R1))
  ae2(qrc <- qr.coef(qr.A, b), with(e.qr.A, solve(R1 %*% P2., t(qrQ)) %*% b))
  ae2(qrf <- qr.fitted(qr.A, b), with(e.qr.A, tcrossprod(qrQ) %*% b))
  ae2(qrr <- qr.resid(qr.A, b), b - qrf)
  ae2(qrq <- qr.qy(qr.A, b), with(E.qr.A, P1. %*% Q %*% b))
  ae2(qr.qty(qr.A, qrq), b)
})

## Sparse and dense computations should agree here
qr.Am <- qr(as(A, "matrix")) # <=> qr.default(A)
stopifnot(exprs = {
  ae2(qrX, qr.X(qr.Am))
  ae2(qrc, qr.coef(qr.Am, b))
  ae2(qrf, qr.fitted(qr.Am, b))
  ae2(qrr, qr.resid(qr.Am, b))
})

```

sparseVector

`classsparseVector`

`sparseVector(x, i, length)`

`x` `"nsparseVector"`

`i` `xTRUE`

`length`

`xdrop0()`

`classsparseVector`

`sparseMatrix()``sparseVector`

```
str(sv <- sparseVector(x = 1:10, i = sample(999, 10), length=1000))
```

```
sx <- c(0,0,3, 3.2, 0,0,0,-3:1,0,0,2,0,0,5,0,0)
```

```
ss <- as(sx, "sparseVector")
```

```
stopifnot(identical(ss,
```

```
  sparseVector(x = c(2, -1, -2, 3, 1, -3, 5, 3.2),
```

```
    i = c(15L, 10:9, 3L,12L,8L,18L, 4L), length = 20L)))
```

```
(ns <- sparseVector(i= c(7, 3, 2), length = 10))
```

```
stopifnot(identical(ns,
```

```
  new("nsparseVector", length = 10, i = c(2, 3, 7))))
```

sparseVector-class

```
"sparseVector""dsparseVector""isparsedVector""lsparseVector""nsparseVector"
"zsparseVector"d*1*n*
```

```
length "numeric"length"numeric""integer".Machine$integer.max
i "numeric"NA
  "integer""numeric"
x "nsparseVector""numeric""dsparseVector""logical""lsparseVector"
```

```
signature(x = "sparseVector")length
signature(object = "sparseVector")show".prSpVector"." "
  options(max.print)
signature(x = "sparseVector", mode = "character")as(x, "vector")
coerce
signature(from = "sparseVector", to = "sparseMatrix")
signature(from = "sparseMatrix", to = "sparseVector")sparseMatrix
signature(x = "sparseVector", value = "integer")sparseMatrix
signature(x = "sparseVector")headhead(x,n)n >= 1x[1:n]
signature(x = "sparseVector")head
signature(x = "sparseVector")toeplitz(x)n × nxn = length(x)
signature(x = "sparseVector")x(x, times, length.out, each,...)
signature(x = "nsparseVector")
signature(x = "lsparseVector")
signature(e1 = "sparseVector", e2 = "*")Ops
signature(x = "sparseVector")Summary
(x = "sparseVector")
(x = "nsparseVector")logical"nsparseVector"xxNANaN
signature(x = "sparseVectors")round()
c.sparseVector()"sparseVector"
```

```
sparseVector()as(*, "sparseVector")
```

```

getClass("sparseVector")
getClass("dsparseVector")

sx <- c(0,0,3, 3.2, 0,0,0,-3:1,0,0,2,0,0,5,0,0)
(ss <- as(sx, "sparseVector"))

ix <- as.integer(round(sx))
(is <- as(ix, "sparseVector")) ## an "isparsedVector" (!)
(ns <- sparseVector(i= c(7, 3, 2), length = 10)) # "nsparsedVector"
## rep() works too:
(ri <- rep(is, length.out= 25))

## Using `dim<-` as in base R :
r <- ss
dim(r) <- c(4,5) # becomes a sparse Matrix:
r
## or coercion (as as.matrix() in base R):
as(ss, "Matrix")
stopifnot(all(ss == print(as(ss, "CsparseMatrix"))))

## currently has "non-structural" FALSE -- printing as ":"
(lis <- is & FALSE)
(nn <- is[lis == 0]) # all "structural" FALSE

## NA-case
sN <- sx; sN[4] <- NA
(svN <- as(sN, "sparseVector"))

v <- as(c(0,0,3, 3.2, rep(0,9),-3,0,-1, rep(0,20),5,0),
        "sparseVector")
v <- rep(rep(v, 50), 5000)
set.seed(1); v[sample(v@i, 1e6)] <- 0
str(v)
system.time(for(i in 1:4) hv <- head(v, 1e6))
## user system elapsed
## 0.033 0.000 0.032
system.time(for(i in 1:4) h2 <- v[1:1e6])
## user system elapsed
## 1.317 0.000 1.319

stopifnot(identical(hv, h2),
          identical(is | FALSE, is != 0),
          validObject(svN), validObject(lis), as.logical(is.na(svN[4])),
          identical(is^2 > 0, is & TRUE),
          all(!lis), !any(lis), length(nn@i) == 0, !any(nn), all(!nn),
          sum(lis) == 0, !prod(lis), range(lis) == c(0,0))

## create and use the t(.) method:
t(x20 <- sparseVector(c(9,3:1), i=c(1:2,4,7), length=20))
(T20 <- toeplitz(x20))
stopifnot(is(T20, "symmetricMatrix"), is(T20, "sparseMatrix"),
          identical(unname(as.matrix(T20)),
                    toeplitz(as.vector(x20))))

## c() method for "sparseVector" - also available as regular function
(c1 <- c(x20, 0,0,0, -10*x20))

```

```

(c2 <- c(ns, is, FALSE))
(c3 <- c(ns, !ns, TRUE, NA, FALSE))
(c4 <- c(ns, rev(ns)))
## here, c() would produce a list {not dispatching to c.sparseVector()}
(c5 <- c.sparseVector(0,0, x20))

## checking (consistency)
.v <- as.vector
.s <- function(v) as(v, "sparseVector")
stopifnot(exprs = {
  all.equal(c1, .s(c(.v(x20), 0,0,0, -10*.v(x20))), tol = 0)
  all.equal(c2, .s(c(.v(ns), .v(is), FALSE)), tol = 0)
  all.equal(c3, .s(c(.v(ns), !.v(ns), TRUE, NA, FALSE)), tol = 0)
  all.equal(c4, .s(c(.v(ns), rev(.v(ns)))), tol = 0,
    check.class = FALSE)
  all.equal(c5, .s(c(0,0, .v(x20))), tol = 0)
})

```

spMatrix

[TsparseMatrix](#)
[sparseMatrix\(\)](#)

spMatrix(nrow, ncol, i = integer(0L), j = integer(0L), x = double(0L))

nrowncol
ij TRUE
x ij

[TsparseMatrixgeneralMatrix](#)

$MM[i[k], j[k]] == x[k]k = 1, 2, \dots, n, n = \text{length}(i)M[i', j'] == 0(i', j')$

[Matrix\(*, sparse=TRUE\)sparseMatrixspMatrix\(\)CsparseMatrixbdiagDiagonal](#)
[TsparseMatrix](#)

```

## simple example
A <- spMatrix(10,20, i = c(1,3:8),
                  j = c(2,9,6:10),
                  x = 7 * (1:7))
A # a "dgTMatrix"
summary(A)
str(A) # note that *internally* 0-based indices (i,j) are used

L <- spMatrix(9, 30, i = rep(1:9, 3), 1:27,

```

```

      (1:27) %% 4 != 1)
L # an "lgTMatrix"

## A simplified predecessor of Matrix' rsparsematrix() function :

rSpMatrix <- function(nrow, ncol, nnz,
                      rand.x = function(n) round(rnorm(nnz), 2))
{
  ## Purpose: random sparse matrix
  ## -----
  ## Arguments: (nrow,ncol): dimension
  ##           nnz : number of non-zero entries
  ##           rand.x: random number generator for 'x' slot
  ## -----
  ## Author: Martin Maechler, Date: 14.-16. May 2007
  stopifnot((nnz <- as.integer(nnz)) >= 0,
            nrow >= 0, ncol >= 0, nnz <= nrow * ncol)
  spMatrix(nrow, ncol,
            i = sample(nrow, nnz, replace = TRUE),
            j = sample(ncol, nnz, replace = TRUE),
            x = rand.x(nnz))
}

M1 <- rSpMatrix(100000, 20, nnz = 200)
summary(M1)

```

subassign-methods

"[<-"

```

matrixx[...] <- valstopclassvalxx"lsparseMatrix"valx"nsparseMatrix"valTRUEFALSE
vallogicalNATRUE

```

[~methods"Matrix"indexExtract

```

set.seed(101)
(a <- m <- Matrix(round(rnorm(7*4),2), nrow = 7))

a[] <- 2.2 # <- replaces **every** entry
a
## as do these:
a[,] <- 3 ; a[TRUE,] <- 4

m[2, 3] <- 3.14 # simple number
m[3, 3:4]<- 3:4 # simple numeric of length 2

## sub matrix assignment:
m[-(4:7), 3:4] <- cbind(1,2:4) #-> upper right corner of 'm'
m[3:5, 2:3] <- 0
m[6:7, 1:2] <- Diagonal(2)
m

## rows or columns only:
m[1,] <- 10
m[,2] <- 1:7
m[-(1:6), ] <- 3:0 # not the first 6 rows, i.e. only the 7th
as(m, "sparseMatrix")

```

subscript-methods

"["

[<--methods"Matrix"Extract

```

str(m <- Matrix(round(rnorm(7*4),2), nrow = 7))
stopifnot(identical(m, m[]))
m[2, 3] # simple number
m[2, 3:4] # simple numeric of length 2
m[2, 3:4, drop=FALSE] # sub matrix of class 'dgeMatrix'
## rows or columns only:
m[1,] # first row, as simple numeric vector
m[,1:2] # sub matrix of first two columns

showMethods("[", inherited = FALSE)

```

symmetricMatrix-class

```
"symmetricMatrix"
```

```
as(*, "symmetricMatrix")
```

```
Dim, Dimnames MatrixDimnames
```

```
factors MatrixFactorization
```

```
uplo "U""L"
```

```
"Matrix"
```

```
signature(object = "symmetricMatrix")dimnamesDimnames
```

```
signature(object = "symmetricMatrix")TRUE
```

```
symmetricMatrix_validate()getValidity(getClass("symmetricMatrix"))
```

[dimnames](#)

```
Dimnameslist(NULL, <character>)dimnames\(\)
```

[isSymmetrictriangularMatrixdsyMatrixlsCMatrix](#)

```
## An example about the symmetric Dimnames:
```

```
sy <- sparseMatrix(i= c(2,4,3:5), j= c(4,7:5,5), x = 1:5, dims = c(7,7),  
                  symmetric=TRUE, dimnames = list(NULL, letters[1:7]))
```

```
sy # shows symmetrical dimnames
```

```
sy@Dimnames # internally only one part is stored
```

```
dimnames(sy) # both parts - as sy *is* symmetrical
```

```
showClass("symmetricMatrix")
```

```
## The names of direct subclasses:
```

```
scl <- getClass("symmetricMatrix")@subclasses
```

```
directly <- sapply(lapply(scl, slot, "by"), length) == 0
```

```
names(scl)[directly]
```

```
## Methods -- applicable to all subclasses above:
```

```
showMethods(classes = "symmetricMatrix")
```

`symmpart-methods`

```
symmpart(x)(x + t(x))/2skewpart(x)(x - t(x))/2x
```

```
x == symmpart(x) + skewpart(x)NA
```

```
symmpart(x)
```

```
skewpart(x)
```

```
x          "matrix"Matrix
```

```
showMethods(symmpart)
```

```
NULLNULL
```

```
symmpart(x)symmetricMatrixdiagonalMatrixxMatrix
```

```
skewpart(x)generalMatrixsymmetricMatrixdiagonalMatrixxMatrix
```

```
isSymmetric
```

```
m <- Matrix(1:4, 2,2)
```

```
symmpart(m)
```

```
skewpart(m)
```

```
stopifnot(all(m == symmpart(m) + skewpart(m)))
```

```
dn <- dimnames(m) <- list(row = c("r1", "r2"), col = c("var.1", "var.2"))
```

```
stopifnot(all(m == symmpart(m) + skewpart(m)))
```

```
colnames(m) <- NULL
```

```
stopifnot(all(m == symmpart(m) + skewpart(m)))
```

```
dimnames(m) <- unname(dn)
```

```
stopifnot(all(m == symmpart(m) + skewpart(m)))
```

```
## investigate the current methods:
```

```
showMethods(skewpart, include = TRUE)
```

triangularMatrix-class

```
"triangularMatrix"nrow == ncol

uplo "character"
diag "character""U""N"diag"U"denseMatrixxdiag
DimDimnames "integer"NULLMatrix

"Matrix"

triangularMatrix_validity()
SchurisSymmetricas()coerceTriangularMatrix

isTriangular()symmetricMatrixdtrMatrixltCMatrix"triangularMatrix"

showClass("triangularMatrix")

## The names of direct subclasses:
scl <- getClass("triangularMatrix")@subclasses
directly <- sapply(lapply(scl, slot, "by"), length) == 0
names(scl)[directly]

(m <- matrix(c(5,1,0,3), 2))
as(m, "triangularMatrix")
```

TsparseMatrix-class

```
"TsparseMatrix"showClass("TsparseMatrix")

DimDimnames "Matrix"
i "integer"0:(nrow(.)-1)
j "integer"i0:(ncol(.)-1)(i,j)dgTMatrix

"sparseMatrix""Matrix""sparseMatrix"
```

```
"["[-methods
```

```
CsparseMatrixCsparseMatrix
```

```
new(.)spMatrixsparseMatrix(*, repr="T")TsparseMatrix $x_k(i_k, j_k)$ dgTMatrix"  
%%crossprodTsparseMatrixTsparseMatrixCsparseMatrix
```

```
sparseMatrixdgTMatrix
```

```
showClass("TsparseMatrix")  
## or just the subclasses' names  
names(getClass("TsparseMatrix")@subclasses)  
  
T3 <- spMatrix(3,4, i=c(1,3:1), j=c(2,4:2), x=1:4)  
T3 # only 3 non-zero entries, 5 = 1+4 !
```

```
unpackedMatrix-class "unpackedMatrix"
```

```
"unpackedMatrix"m*nmn"[dln]geMatrix""[dln]syMatrix""[dln]trMatrix""dpoMatrix"
```

```
DimDimnames Matrix
```

```
"denseMatrix""Matrix""denseMatrix"
```

```
signature(x = "unpackedMatrix")  
signature(x = "unpackedMatrix")  
signature(object = "unpackedMatrix")  
signature(object = "unpackedMatrix")  
signature(object = "unpackedMatrix")  
signature(x = "unpackedMatrix")  
signature(x = "unpackedMatrix")  
signature(x = "unpackedMatrix")
```

```
packunpack"packedMatrix""dsyMatrix""ltrMatrix"
```

```
showClass("unpackedMatrix")  
showMethods(classes = "unpackedMatrix")
```

updown-methods

k

$$P_1 A P_1' = L_1 D L_1' = L L'$$

kC

$$P_1 (A + s C C') P_1' = \tilde{L}_1 \tilde{D} \tilde{L}_1' = \tilde{L} \tilde{L}'$$

$s = 1 \quad s = -1$

updown(update, C, L)

update	TRUEFALSE"+""-""L
C	Matrixtcrossprod(C) L
L	dCHMsimpldCHMsuper

[LdCHMsimpl](#)

[dCHMsimpldCHMsuperupdateupdown\(update = TRUE\)](#)

[Cholesky](#)

```
m <- sparseMatrix(i = c(3, 1, 3:2, 2:1), p = c(0:2, 4, 4, 6), x = 1:6,
  dimnames = list(LETTERS[1:3], letters[1:5]))
uc0 <- Cholesky(A <- crossprod(m) + Diagonal(5))
uc1 <- updown("+", Diagonal(5, 1), uc0)
uc2 <- updown("-", Diagonal(5, 1), uc1)
stopifnot(all.equal(uc0, uc2))
```

USCounties

```
data(USCounties)
```

```
3111 × 3111dsCMatrix
```

```
usc_q.GALhttp://sal.uiuc.edu/weights/zips/usc.zipread.gal
```

```
nb2listwsimilar.listwlistwdsTMatrixas_dsTMatrix_listwdsCMatrix
```

```
data(USCounties, package = "Matrix")
(n <- ncol(USCounties))
I <- .symDiagonal(n)

set.seed(1)
r <- 50L
rho <- 1 / runif(r, 0, 0.5)

system.time(MJ0 <- sapply(rho, function(mult)
  determinant(USCounties + mult * I, logarithm = TRUE)$modulus))

## Can be done faster by updating the Cholesky factor:

C1 <- Cholesky(USCounties, Imult = 2)
system.time(MJ1 <- sapply(rho, function(mult)
  determinant(update(C1, USCounties, mult), sqrt = FALSE)$modulus))
stopifnot(all.equal(MJ0, MJ1))

C2 <- Cholesky(USCounties, super = TRUE, Imult = 2)
system.time(MJ2 <- sapply(rho, function(mult)
  determinant(update(C2, USCounties, mult), sqrt = FALSE)$modulus))
stopifnot(all.equal(MJ0, MJ2))
```

wrld_1deg

```
data(wrld_1deg)
```

```
15260 × 15260dsCMatrix
```

```
RgshhsSobj_SpatialGridoverSpatialPolygonsSpatialGridSpatialPixelsdneareigh  
sqrt(2)  
nb2listwsimilar.listwlistwdsTMatrixas_dsTMatrix_listwdsCMatrix
```

```
data(wrld_1deg, package = "Matrix")  
(n <- ncol(wrld_1deg))  
I <- .symDiagonal(n)  
  
doExtras <- interactive() || nzchar(Sys.getenv("R_MATRIX_CHECK_EXTRA"))  
set.seed(1)  
r <- if(doExtras) 20L else 3L  
rho <- 1 / runif(r, 0, 0.5)  
  
system.time(MJ0 <- sapply(rho, function(mult)  
  determinant(wrld_1deg + mult * I, logarithm = TRUE)$modulus))  
  
## Can be done faster by updating the Cholesky factor:  
  
C1 <- Cholesky(wrld_1deg, Imult = 2)  
system.time(MJ1 <- sapply(rho, function(mult)  
  determinant(update(C1, wrld_1deg, mult), sqrt = FALSE)$modulus))  
stopifnot(all.equal(MJ0, MJ1))  
  
C2 <- Cholesky(wrld_1deg, super = TRUE, Imult = 2)  
system.time(MJ2 <- sapply(rho, function(mult)  
  determinant(update(C2, wrld_1deg, mult), sqrt = FALSE)$modulus))  
stopifnot(all.equal(MJ0, MJ2))
```


boot

`abc.ci`

```
abc.ci(data, statistic, index=1, strata=rep(1, n), conf=0.95,  
       eps=0.001/n, ...)
```

```
data  
statistic      statisticabc.ci...{}  
index          statistic  
strata  
conf  
eps  
...            statisticstatisticabc.ci
```

```
abcnon2*n+2+2*length(conf)n
```

```
length(conf)
```

[boot.ci](#)


```

# 90% and 95% confidence intervals for the correlation
# coefficient between the columns of the bigcity data

abc.ci(bigcity, corr, conf=c(0.90,0.95))

# A 95% confidence interval for the difference between the means of
# the last two samples in gravity
mean.diff <- function(y, w)
{
  gp1 <- 1:table(as.numeric(y$series))[1]
  sum(y[gp1, 1] * w[gp1]) - sum(y[-gp1, 1] * w[-gp1])
}
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
## IGNORE_RDIFF_BEGIN
abc.ci(grav1, mean.diff, strata = grav1$series)
## IGNORE_RDIFF_END

```

acme

acme

acme

month

market

acme

aids

aids

aids

year

quarter

delay delay2

dud

time

y

aircondit

airconditaircondit7

aircondit

hours

amis

amis

amis

speed
period
warning
pair

aml

aml

aml

time
cens time
group

amlgroupx

beaver

beaver"ts""data.frame"

beaver

day
time
temp
activ 1

bigcity

bigcity
city
citybigcity

bigcity

u
x

boot

R

```
boot(data, statistic, R, sim = "ordinary", stype = c("i", "f", "w"),
      strata = rep(1,n), L = NULL, m = 0, weights = NULL,
      ran.gen = function(d, p) d, mle = NULL, simple = FALSE, ...,
      parallel = c("no", "multicore", "snow"),
      ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

data

statistic sim = "parametric" statistic ran.gen statistic statistic...

R R

sim "ordinary""parametric""balanced""permutation""antithetic"
 "ordinary""balanced"

stype statistic"i""f""w" sim = "parametric"

strata sim = "parametric" strata

L sim"antithetic"empinfstype"w"

m sim"ordinary"mstatistic

weights dataweightsweightsRnrow(weights)sim"ordinary""balanced"

ran.gen sim = "parametric" ran.gen statistic ran.gendata statistic sim =
 "parametric" ran.gen

mle ran.gen mle ran.gen

simple TRUE sim = "ordinary", stype = "i", n = 0 nR simple = TRUE

... statistic statistic statistic boot XFUN

parallel "boot.parallel""no"

ncpus

cl parallel = "snow" boot

statisticboot

ran.genmle

"boot"

t0 statisticdata

t sum(R) statistic

R Rboot

```

data          databoot
seed          .Random.seedboot
statistic     statisticboot
sim
stype         boot
call          boot
strata        bootsim"parametric"
weights       bootsim"ordinary""balanced"
pred.i        m > 0m0sim"ordinary"
L             sim"antithetic"stype"w"Lsim"antithetic"
ran.gen       sim"parametric"sim
mle           bootsim"parametric"sim

cplotprint

```

```

parallel = "multicore"

parallel = "snow"statisticstatisticjack.after.bootstatisticparallel = "snow"
parallel = "multicore"

bootsimple = TRUEsim = "parametric"statisticcensboot(sim = "wierd")tsbootsim ==
"fixed"sim == "geom"ran.gen

parallel = "multicore"parallel = "snow"RNGkind("L'Ecuyer-CMRG")ncpusparallel =
"multicore"parallel::mc.reset.stream()mclapply

```

```

boot.arrayboot.cicensbootempinfjack.after.boottilt.boottsboot

```

```

# Usual bootstrap of the ratio of means using the city data
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
boot(city, ratio, R = 999, stype = "w")

# Stratified resampling for the difference of means. In this
# example we will look at the difference of means between the final
# two series in the gravity data.
diff.means <- function(d, f)
{
  n <- nrow(d)
  gp1 <- 1:table(as.numeric(d$series))[1]
  m1 <- sum(d[gp1,1] * f[gp1])/sum(f[gp1])
  m2 <- sum(d[-gp1,1] * f[-gp1])/sum(f[-gp1])
  ss1 <- sum(d[gp1,1]^2 * f[gp1]) - (m1 * m1 * sum(f[gp1]))
  ss2 <- sum(d[-gp1,1]^2 * f[-gp1]) - (m2 * m2 * sum(f[-gp1]))
  c(m1 - m2, (ss1 + ss2)/(sum(f) - 2))
}
grav1 <- gravity[as.numeric(gravity[,2]) >= 7,]
boot(grav1, diff.means, R = 999, stype = "f", strata = grav1[,2])

# In this example we show the use of boot in a prediction from
# regression based on the nuclear data. This example is taken
# from Example 6.8 of Davison and Hinkley (1997). Notice also
# that two extra arguments to 'statistic' are passed through boot.
nuke <- nuclear[, c(1, 2, 5, 7, 8, 10, 11)]
nuke.lm <- glm(log(cost) ~ date+log(cap)+ne+ct+log(cum.n)+pt, data = nuke)
nuke.diag <- glm.diag(nuke.lm)
nuke.res <- nuke.diag$res * nuke.diag$sd
nuke.res <- nuke.res - mean(nuke.res)

# We set up a new data frame with the data, the standardized
# residuals and the fitted values for use in the bootstrap.
nuke.data <- data.frame(nuke, resid = nuke.res, fit = fitted(nuke.lm))

# Now we want a prediction of plant number 32 but at date 73.00
new.data <- data.frame(cost = 1, date = 73.00, cap = 886, ne = 0,
  ct = 0, cum.n = 11, pt = 1)
new.fit <- predict(nuke.lm, new.data)

nuke.fun <- function(dat, inds, i.pred, fit.pred, x.pred)
{
  lm.b <- glm(fit+resid[inds] ~ date+log(cap)+ne+ct+log(cum.n)+pt,
    data = dat)
  pred.b <- predict(lm.b, x.pred)
  c(coef(lm.b), pred.b - (fit.pred + dat$resid[i.pred]))
}

nuke.boot <- boot(nuke.data, nuke.fun, R = 999, m = 1,
  fit.pred = new.fit, x.pred = new.data)
# The bootstrap prediction squared error would then be found by
mean(nuke.boot$t[, 8]^2)
# Basic bootstrap prediction limits would be
new.fit - sort(nuke.boot$t[, 8])[c(975, 25)]

# Finally a parametric bootstrap. For this example we shall look

```

```

# at the air-conditioning data. In this example our aim is to test
# the hypothesis that the true value of the index is 1 (i.e. that
# the data come from an exponential distribution) against the
# alternative that the data come from a gamma distribution with
# index not equal to 1.
air.fun <- function(data) {
  ybar <- mean(data$hours)
  para <- c(log(ybar), mean(log(data$hours)))
  ll <- function(k) {
    if (k <= 0) 1e200 else lgamma(k)-k*(log(k)-1-para[1]+para[2])
  }
  khat <- nlm(ll, ybar^2/var(data$hours))$estimate
  c(ybar, khat)
}

air.rg <- function(data, mle) {
  # Function to generate random exponential variates.
  # mle will contain the mean of the original data
  out <- data
  out$hours <- rexp(nrow(out), 1/mle)
  out
}

air.boot <- boot(aircondit, air.fun, R = 999, sim = "parametric",
  ran.gen = air.rg, mle = mean(aircondit$hours))

# The bootstrap p-value can then be approximated by
sum(abs(air.boot$t[,2]-1) > abs(air.boot$t[0][2]-1))/(1+air.boot$R)

```

```
boot.array
```

```
bootcensboottilt.boot
```

```
boot.array(boot.out, indices)
```

```
boot.out      "boot"
indices       indices=FALSEboot.outtsbootindices=TRUE
```

```
.Random.seedfreq.array
censbootboot.out$sim"parametric"boot"model"tsboottsboot
```

```
boot.out$Rnnboot.out$dataindicesFALSEindicesTRUE
```

```
.Random.seedboot.out$seed
```


[bootcensbootfreq.arraytilt.boottsboot](#)

```
# A frequency array for a nonparametric bootstrap
city.boot <- boot(city, corr, R = 40, stype = "w")
boot.array(city.boot)

perm.cor <- function(d,i) cor(d$x,d$u[i])
city.perm <- boot(city, perm.cor, R = 40, sim = "permutation")
boot.array(city.perm, indices = TRUE)
```

[boot.ci](#)

```
boot.ci(boot.out, conf = 0.95, type = "all",
        index = 1:min(2,length(boot.out$t0)), var.t0 = NULL,
        var.t = NULL, t0 = NULL, t = NULL, L = NULL,
        h = function(t) t, hdot = function(t) rep(1,length(t)),
        hinv = function(t) t, ...)
```

boot.out	"boot"
conf	
type	c("norm","basic", "stud", "perc", "bca")"all"
index	indexboot.out\$t0boot.out\$tvar.t0var.tindexboot.out\$t0
var.t0	length(index)var.t0boot.out\$t0[index[2]]var.t0var.t0var.t0 var(t)hvar.t0
var.t	boot.out\$Rlength(index)var.tboot.out\$t[,index[2]]hvar.t
t0	boot.out\$t0[index[1]]t0thhdothinv
t	boot.out\$Rt0tt0tvar.t0var.tboot.out\$t[,index]
L	hLth(t)hdotLempinf
h	h(t)hinvh(c(t1,t2,t3))c(h(t1),h(t2),h(t3))
hdot	hhh(t0)h(t)h
hinv	hhh(t)hhinv
...	boot.out\$statisticLlempinfboot.out\$statistic

bootboot.out

"bootci"

R

t0

call boot.ci
 typebootci

normal

basic

student

percent

bca

[abc.cibootempinfnorm.ci](#)

```
# confidence intervals for the city data
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
city.boot <- boot(city, ratio, R = 999, stype = "w", sim = "ordinary")
boot.ci(city.boot, conf = c(0.90, 0.95),
         type = c("norm", "basic", "perc", "bca"))

# studentized confidence interval for the two sample
# difference of means problem using the final two series
# of the gravity data.
diff.means <- function(d, f)
{
  n <- nrow(d)
  gp1 <- 1:table(as.numeric(d$series))[1]
  m1 <- sum(d[gp1,1] * f[gp1])/sum(f[gp1])
  m2 <- sum(d[-gp1,1] * f[-gp1])/sum(f[-gp1])
  ss1 <- sum(d[gp1,1]^2 * f[gp1]) - (m1 * m1 * sum(f[gp1]))
  ss2 <- sum(d[-gp1,1]^2 * f[-gp1]) - (m2 * m2 * sum(f[-gp1]))
  c(m1 - m2, (ss1 + ss2)/(sum(f) - 2))
}
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav1.boot <- boot(grav1, diff.means, R = 999, stype = "f",
                  strata = grav1[,2])
boot.ci(grav1.boot, type = c("stud", "norm"))

# Nonparametric confidence intervals for mean failure time
# of the air-conditioning data as in Example 5.4 of Davison
# and Hinkley (1997)
mean.fun <- function(d, i)
{
  m <- mean(d$hours[i])
```

```

      n <- length(i)
      v <- (n-1)*var(d$hours[i])/n^2
      c(m, v)
    }
    air.boot <- boot(aircondit, mean.fun, R = 999)
    boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"))

    # Now using the log transformation
    # There are two ways of doing this and they both give the
    # same intervals.

    # Method 1
    boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"),
            h = log, hdot = function(x) 1/x)

    # Method 2
    vt0 <- air.boot$t0[2]/air.boot$t0[1]^2
    vt <- air.boot$t[, 2]/air.boot$t[, 1]^2
    boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"),
            t0 = log(air.boot$t0[1]), t = log(air.boot$t[,1]),
            var.t0 = vt0, var.t = vt)

```

brambles

brambles

brambles

x

y

age 012

breslow

breslow

breslow

breslow

age

smoke

n

y

ns smoke*n

calcium

calcium

calcium

time

cal

cane

cane

cane

n

r

x

var

block

capability

capability

capability

y

catsM

catsM
catsMcatsMASS

catsM

Sex FMM
Bwt
Hwt

cats

cav

cav
caveolae.dat

cav

x
y

cd4

cd4

cd4

baseline

oneyear

cd4.nested

cd4

cd4

censboot

```
censboot(data, statistic, R, F.surv, G.surv, strata = matrix(1,n,2),
          sim = "ordinary", cox = NULL, index = c(1, 2), ...,
          parallel = c("no", "multicore", "snow"),
          ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

```
data          sim = "weird" data index
statistic     ...sim = "weird" statistic data sim = "weird" data sim = "weird" strata
              statistic
R
F.surv        survfitsim = "ordinary" sim = "model" cox
G.surv        survfitsurvfitsim = "cond" sim = "model" cox
strata        survfitsim = "weird" sim = "ordinary" strata
sim           "ordinary" "model" "ordinary" cox "weird" cox "cond"
cox           coxphF.survsurvfit(cox)
index         data
...           statistic statistic statistic censboot XFUN
parallel ncpus cl
              boot
```

```
InfInf
sim = "model" sim = "cond"
```

```
sim "model" "cond" InfInf
parallel = "snow" cl library(survival)
```

```
"boot"

t0            statistic
t             statistic
R
sim           sim "model" cox "ordinary"
data          datasim = "weird"
seed          .Random.seed censboot
statistic     statistic
strata        sim = "ordinary" sim = "weird"
call          censboot
```


bootcoxphsurvfit

```
library(survival)
# Example 3.9 of Davison and Hinkley (1997) does a bootstrap on some
# remission times for patients with a type of leukaemia. The patients
# were divided into those who received maintenance chemotherapy and
# those who did not. Here we are interested in the median remission
# time for the two groups.
data(aml, package = "boot") # not the version in survival.
aml.fun <- function(data) {
  surv <- survfit(Surv(time, cens) ~ group, data = data)
  out <- NULL
  st <- 1
  for (s in 1:length(surv$strata)) {
    inds <- st:(st + surv$strata[s]-1)
    md <- min(surv$time[inds[1-surv$surv[inds] >= 0.5]])
    st <- st + surv$strata[s]
    out <- c(out, md)
  }
  out
}
aml.case <- censboot(aml, aml.fun, R = 499, strata = aml$group)

# Now we will look at the same statistic using the conditional
# bootstrap and the weird bootstrap. For the conditional bootstrap
# the survival distribution is stratified but the censoring
# distribution is not.

aml.s1 <- survfit(Surv(time, cens) ~ group, data = aml)
aml.s2 <- survfit(Surv(time-0.001*cens, 1-cens) ~ 1, data = aml)
aml.cond <- censboot(aml, aml.fun, R = 499, strata = aml$group,
  F.surv = aml.s1, G.surv = aml.s2, sim = "cond")

# For the weird bootstrap we must redefine our function slightly since
# the data will not contain the group number.
aml.fun1 <- function(data, str) {
  surv <- survfit(Surv(data[, 1], data[, 2]) ~ str)
  out <- NULL
  st <- 1
  for (s in 1:length(surv$strata)) {
```

```

        inds <- st:(st + surv$strata[s] - 1)
        md <- min(surv$time[inds[1-surv$surv[inds] >= 0.5]])
        st <- st + surv$strata[s]
        out <- c(out, md)
    }
    out
}
aml.wei <- censboot(cbind(aml$time, aml$cens), aml.fun1, R = 499,
    strata = aml$group, F.surv = aml.s1, sim = "weird")

# Now for an example where a cox regression model has been fitted
# the data we will look at the melanoma data of Example 7.6 from
# Davison and Hinkley (1997). The fitted model assumes that there
# is a different survival distribution for the ulcerated and
# non-ulcerated groups but that the thickness of the tumour has a
# common effect. We will also assume that the censoring distribution
# is different in different age groups. The statistic of interest
# is the linear predictor. This is returned as the values at a
# number of equally spaced points in the range of interest.
data(melanoma, package = "boot")
library(splines)# for ns
mel.cox <- coxph(Surv(time, status == 1) ~ ns(thickness, df=4) + strata(ulcer),
    data = melanoma)
mel.surv <- survfit(mel.cox)
agec <- cut(melanoma$age, c(0, 39, 49, 59, 69, 100))
mel.cens <- survfit(Surv(time - 0.001*(status == 1), status != 1) ~
    strata(agec), data = melanoma)
mel.fun <- function(d) {
    t1 <- ns(d$thickness, df=4)
    cox <- coxph(Surv(d$time, d$status == 1) ~ t1+strata(d$ulcer))
    ind <- !duplicated(d$thickness)
    u <- d$thickness[!ind]
    eta <- cox$linear.predictors[!ind]
    sp <- smooth.spline(u, eta, df=20)
    th <- seq(from = 0.25, to = 10, by = 0.25)
    predict(sp, th)$y
}
mel.str <- cbind(melanoma$ulcer, agec)

# this is slow!
mel.mod <- censboot(melanoma, mel.fun, R = 499, F.surv = mel.surv,
    G.surv = mel.cens, cox = mel.cox, strata = mel.str, sim = "model")
# To plot the original predictor and a 95% pointwise envelope for it
mel.env <- envelope(mel.mod)$point
th <- seq(0.25, 10, by = 0.25)
plot(th, mel.env[1, ], ylim = c(-2, 2),
    xlab = "thickness (mm)", ylab = "linear predictor", type = "n")
lines(th, mel.mod$t0, lty = 1)
matlines(th, t(mel.env), lty = 2)

```

channing

channing

channing

sex "Male""Female"

entry

exit

time time=exit-entry

cens

claridge

claridge

claridge

dnan

hand

cloth

cloth

cloth

x

y

co.transfer

co.transfer

co.transfer

entry

week

coal

coal

coal

date date

control

```
control(boot.out, L = NULL, distn = NULL, index = 1, t0 = NULL,
        t = NULL, bias.adj = FALSE, alpha = NULL, ...)
```

boot.out	boot
L	Lempinfboot.out
distn	smooth.splinebias.adjFALSEsaddle.distn
index	boot.out\$statistic
t0	boot.out\$databias.adjFALSEtboot.out\$t0[index]
t	bias.adjFALSEt0boot.out\$t[,index]
bias.adj	TRUEbias.adjFALSE
alpha	bias.adjFALSE
...	boot.out\$statisticboot.out\$statisticboot.out\$statisticbias.adj TRUEempinfl

bias.adjFALSEttt

bias.adjTRUE

bias.adjFALSE

L	empinf
tL	t
bias	t
var	t
k3	t
quantiles	t
distn	smooth.splinetdistnsaddle.distn

[bootempinfk3.linearlinear.approxsaddle.distnsmooth.splinevar.linear](#)

```
# Use of control variates for the variance of the air-conditioning data
mean.fun <- function(d, i)
{
  m <- mean(d$hours[i])
  n <- nrow(d)
  v <- (n-1)*var(d$hours[i])/n^2
  c(m, v)
}
air.boot <- boot(aircondit, mean.fun, R = 999)
control(air.boot, index = 2, bias.adj = TRUE)
air.cont <- control(air.boot, index = 2)
# Now let us try the variance on the log scale.
air.cont1 <- control(air.boot, t0 = log(air.boot$t0[2]),
                    t = log(air.boot$t[, 2]))
```

corr

corr(d, w = rep(1, nrow(d))/nrow(d))

```
d
w          sum(w)
```

```
d[,1]d[,2]
```

```
cor
```

```
cum3
```

```
cum3(a, b = a, c = a, unbiased = TRUE)
```

```
a
b          aa
c          aa
unbiased
```

```
n/((n-1)*(n-2))nunbiasedFALSE1/nsum((a-mean(a))*(b-mean(b))*(c-mean(c)))
```

`cv.glm`

```
cv.glm(data, glmfit, cost, K)
```

```
data
glmfit      "glm"data
cost        costcost
K           KKKKdata
```

```
Kdatacost
KKKKdelta
```

```
call        cv.glm
K           K
delta
seed        .Random.seedcv.glm
```

```
.Random.seed
```

[glmglm.diagpredict](#)


```

# leave-one-out and 6-fold cross-validation prediction error for
# the mammals data set.
data(mammals, package="MASS")
mammals.glm <- glm(log(brain) ~ log(body), data = mammals)
(cv.err <- cv.glm(mammals, mammals.glm)$delta)
(cv.err.6 <- cv.glm(mammals, mammals.glm, K = 6)$delta)

# As this is a linear model we could calculate the leave-one-out
# cross-validation estimate without any extra model-fitting.
muhat <- fitted(mammals.glm)
mammals.diag <- glm.diag(mammals.glm)
(cv.err <- mean((mammals.glm$y - muhat)^2/(1 - mammals.diag$h)^2))

# leave-one-out and 11-fold cross-validation prediction error for
# the nodal data set. Since the response is a binary variable an
# appropriate cost function is
cost <- function(r, pi = 0) mean(abs(r-pi) > 0.5)

nodal.glm <- glm(r ~ stage+xray+acid, binomial, data = nodal)
(cv.err <- cv.glm(nodal, nodal.glm, cost, K = nrow(nodal))$delta)
(cv.11.err <- cv.glm(nodal, nodal.glm, cost, K = 11)$delta)

```

darwin

darwin

darwin

y

dogs

dogs

dogs

downs.bc

downs.bc

downs.bc

age

m

r

ducks

ducks

ducks

plumage
behaviour

F_2

EEF.profile

```
EEF.profile(y, tmin = min(y) + 0.1, tmax = max(y) - 0.1, n.t = 25,  
            u = function(y, t) y - t)  
EL.profile(y, tmin = min(y) + 0.1, tmax = max(y) - 0.1, n.t = 25,  
           u = function(y, t) y - t)
```

```
y  
tmin      min(y)  
tmax      max(y)  
n.t       tmin  
u
```

bootboot<https://artowen.su.domains/empirical/>

n.tEL.profileEEF.profile

empinf

```
empinf(boot.out = NULL, data = NULL, statistic = NULL,  
       type = NULL, stype = NULL, index = 1, t = NULL,  
       strata = rep(1, n), eps = 0.001, ...)
```

boot.out	boottype"reg"datastatisticstypestrataboot.outempinf
data	boot.outboot.outdataboot.out\$data
statistic	stypeempinfstatisticboot.outboot.out
type	type"inf""jack""pos""reg"tttype"reg"boot.outtstype"w"type"inf" boot.out"reg""jack"type"reg"boot.out"inf"stype"w"
stype	statistic"w""f""i"boot.outstypeboot.out\$stype"w"type"inf"stype "w"
index	statistic
t	boot.out\$Rttyperegboot.out\$t[,index]
strata	boot.outstrataboot.out\$strata
eps	type"inf"epsdata
...	statisticstatistic

```
type"inf"stype"w"type"jack"type"pos"type"reg"statisticboot.array  
empinf
```

```
statisticdata
```

```
empinfname = value
```

```
bootboot.arrayboot.cicontroljack.after.bootlinear.approxvar.linear
```

```

# The empirical influence values for the ratio of means in
# the city data.
ratio <- function(d, w) sum(d$x *w)/sum(d$u*w)
empinf(data = city, statistic = ratio)
city.boot <- boot(city, ratio, 499, stype="w")
empinf(boot.out = city.boot, type = "reg")

# A statistic that may be of interest in the difference of means
# problem is the t-statistic for testing equality of means. In
# the bootstrap we get replicates of the difference of means and
# the variance of that statistic and then want to use this output
# to get the empirical influence values of the t-statistic.
grav1 <- gravity[as.numeric(gravity[,2]) >= 7,]
grav.fun <- function(dat, w) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2] - mns[1], s2hat)
}

grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w",
  strata = grav1[, 2])

# Since the statistic of interest is a function of the bootstrap
# statistics, we must calculate the bootstrap replicates and pass
# them to empinf using the t argument.
grav.z <- (grav.boot$t[,1]-grav.boot$t0[1])/sqrt(grav.boot$t[,2])
empinf(boot.out = grav.boot, t = grav.z)

```

envelope

```
envelope(boot.out = NULL, mat = NULL, level = 0.95, index = 1:ncol(mat))
```

```

boot.out      "boot"boot.out$t
mat           boot.outboot.out$t
level
index         mat

mat

```

```

point
overall      point
k.pt
err.pt
k.ov
err.ov
err.nom

```

[bootboot.ci](#)

```

# Testing whether the final series of measurements of the gravity data
# may come from a normal distribution. This is done in Examples 4.7
# and 4.8 of Davison and Hinkley (1997).
grav1 <- gravity$g[gravity$series == 8]
grav.z <- (grav1 - mean(grav1))/sqrt(var(grav1))
grav.gen <- function(dat, mle) rnorm(length(dat))
grav.qqboot <- boot(grav.z, sort, R = 999, sim = "parametric",
                    ran.gen = grav.gen)
grav.qq <- qqnorm(grav.z, plot.it = FALSE)
grav.qq <- lapply(grav.qq, sort)
plot(grav.qq, ylim = c(-3.5, 3.5), ylab = "Studentized Order Statistics",
     xlab = "Normal Quantiles")
grav.env <- envelope(grav.qqboot, level = 0.9)
lines(grav.qq$x, grav.env$point[1, ], lty = 4)
lines(grav.qq$x, grav.env$point[2, ], lty = 4)
lines(grav.qq$x, grav.env$overall[1, ], lty = 1)
lines(grav.qq$x, grav.env$overall[2, ], lty = 1)

```

exp.tilt

```

exp.tilt(L, theta = NULL, t0 = 0, lambda = NULL,
         strata = rep(1, length(L)))

```

L	LLempinf
theta	lambda
t0	
lambda	lambdaexp(lambda * L)lambdathetalambdalambdathetalambdatheta
strata	LL

```

thetap[j]exp(lambda*L[j]/n)nlambda
thetambda
nlambda
min

imp.probimp.quantile

smooth.ftilt.bootexp.tiltsmooth.f

```

```

p          mmthetambda
length(L) mpmlength(L)
p[i,]theta[i]
pweightsboot

lambda     lambdatheta

theta      thetathetambda
theta

```

```

empinfimp.probimp.quantile
optims
smooth.ftilt.boot

```

```

# Example 9.8 of Davison and Hinkley (1997) requires tilting the resampling
# distribution of the studentized statistic to be centred at the observed
# value of the test statistic 1.84. This can be achieved as follows.
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2]-mns[1], s2hat, (mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.L <- empinf(data = grav1, statistic = grav.fun, stype = "w",
  strata = grav1[,2], index = 3, orig = grav.z0[1])
grav.tilt <- exp.tilt(grav.L, grav.z0[3], strata = grav1[,2])
boot(grav1, grav.fun, R = 499, stype = "w", weights = grav.tilt$p,
  strata = grav1[,2], orig = grav.z0[1])

```

```

fir

```

```

fir

```

```

fir

```

count
row
col

freq.array

freq.array(i.array)

i.array bootboot.array

boot.array

frets

frets

frets

l1
b1
l2
b2

glm.diag

glm.diag(glmfit)

glmfit glmfitglm.objectglm()

res
rd
rp
cook
h
sd

[glm.diag.plots](#)glm.diag

[glmglm.diag.plotssummary.glm](#)

glm.diag.plots

glm.diag.plots(glmfit, glmdiat = glm.diag(glmfit), subset = NULL,
 iden = FALSE, labels = NULL, ret = FALSE)

glmfit	glm.objectglm()
glmdiat	glmfitglm.diag
subset	dataglmsubsetglm()glmfitsubset=glm
iden	TRUEidentify()identify()idenFALSE
labels	identify()idenTRUEglmfit
ret	glmdiatFALSE

glm.diag

iden=T

retTRUEglm.diag

split.screen(c(2,2))idenTRUE

[glm.diag.identify](#)

```
# In this example we look at the leukaemia data which was looked at in
# Example 7.1 of Davison and Hinkley (1997)
data(leuk, package = "MASS")
leuk.mod <- glm(time ~ ag-1+log10(wbc), family = Gamma(log), data = leuk)
leuk.diag <- glm.diag(leuk.mod)
glm.diag.plots(leuk.mod, leuk.diag)
```

gravity

gravity

grav

gravitygrav

gravity

g

series

hirose

hirose

hirose

volt
time
cens 1

Imp.Estimates

```
imp.moments(boot.out = NULL, index = 1, t = boot.out$t[, index],  
             w = NULL, def = TRUE, q = NULL)  
imp.prob(boot.out = NULL, index = 1, t0 = boot.out$t0[index],  
          t = boot.out$t[, index], w = NULL, def = TRUE, q = NULL)  
imp.quantile(boot.out = NULL, alpha = NULL, index = 1,  
              t = boot.out$t[, index], w = NULL, def = TRUE, q = NULL)
```

boot.out	"boot"boottilt.bootwboot.outt
alpha	
index	boot.out\$statistict
t0	t0[i]t0[i]imp.probt0
t	boot.outboot.outboot.outt
w	boot.outimp.weights
def	wimp.weightsw
q	wwwqimp.weightsw

alpha	alphaimp.quantile
t0	imp.prob
raw	imp.momentsimp.probrawt0imp.quantilealpha
rat	wraw
reg	t*wwraw

[bootexp.tiltimp.weightssmooth.ftilt.boot](#)

```
# Example 9.8 of Davison and Hinkley (1997) requires tilting the
# resampling distribution of the studentized statistic to be centred
# at the observed value of the test statistic, 1.84. In this example
# we show how certain estimates can be found using resamples taken from
# the tilted distribution.
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2] - mns[1], s2hat, (mns[2] - mns[1] - orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.L <- empinf(data = grav1, statistic = grav.fun, stype = "w",
  strata = grav1[,2], index = 3, orig = grav.z0[1])
grav.tilt <- exp.tilt(grav.L, grav.z0[3], strata = grav1[, 2])
grav.tilt.boot <- boot(grav1, grav.fun, R = 199, stype = "w",
  strata = grav1[, 2], weights = grav.tilt$p,
  orig = grav.z0[1])
# Since the weights are needed for all calculations, we shall calculate
# them once only.
grav.w <- imp.weights(grav.tilt.boot)
grav.mom <- imp.moments(grav.tilt.boot, w = grav.w, index = 3)
grav.p <- imp.prob(grav.tilt.boot, w = grav.w, index = 3, t0 = grav.z0[3])
unlist(grav.p)
grav.q <- imp.quantile(grav.tilt.boot, w = grav.w, index = 3,
  alpha = c(0.9, 0.95, 0.975, 0.99))
as.data.frame(grav.q)
```

`imp.weights`

`pq`

`imp.weights(boot.out, def = TRUE, q = NULL)`

`boot.out` `"boot"boottilt.boot`

`def` `boot.out`

`q` `qboot.out$dataq`

`fprod((q/p)^f)qp`

`boot.out$tboot.out$tq`

`imp.momentsimp.weights`

[bootexp.tiltimp.momentssmooth.ftilt.boot](#)

`inv.logit`

`inv.logit(x)`

`x` `NA`

`exp(x)/(1+exp(x))x-InfInfNANA`

`x`

[logitplogis](#)

islay

islay

islay

theta

jack.after.boot

```
jack.after.boot(boot.out, index = 1, t = NULL, L = NULL,  
                useJ = TRUE, stinf = TRUE, alpha = NULL,  
                main = "", ylab = NULL, ...)
```

boot.out	"boot" boot boot.out\$R
index	boot.out\$statistic
t	boot.out\$Rboot.out\$t[,index]
L	useJFALSEempinfL
useJ	FALSETRUE
stinf	TRUE
alpha	c(0.05, 0.1, 0.16, 0.5, 0.84, 0.9, 0.95)
main	
ylab	alphaylabalpha
...	boot.out\$statisticuseJFALSELempinf

```
useJTRUEuseJFALSEempinf
```

```
bootempinf
```

```
# To draw the jackknife-after-bootstrap plot for the head size data as in
# Example 3.24 of Davison and Hinkley (1997)
frets.fun <- function(data, i) {
  pcorr <- function(x) {
    # Function to find the correlations and partial correlations between
    # the four measurements.
    v <- cor(x)
    v.d <- diag(var(x))
    iv <- solve(v)
    iv.d <- sqrt(diag(iv))
    iv <- - diag(1/iv.d) %*% iv %*% diag(1/iv.d)
    q <- NULL
    n <- nrow(v)
    for (i in 1:(n-1))
      q <- rbind( q, c(v[i, 1:i], iv[i,(i+1):n]) )
    q <- rbind( q, v[n, ] )
    diag(q) <- round(diag(q))
    q
  }
  d <- data[i, ]
  v <- pcorr(d)
  c(v[1,], v[2,], v[3,], v[4,])
}
frets.boot <- boot(log(as.matrix(frets)), frets.fun, R = 999)
# we will concentrate on the partial correlation between head breadth
# for the first son and head length for the second. This is the 7th
# element in the output of frets.fun so we set index = 7
jack.after.boot(frets.boot, useJ = FALSE, stinf = FALSE, index = 7)
```

k3.linear

```
k3.linear(L, strata = NULL)
```

L	empinf
strata	L

L

[empinflinear.approxvar.linear](#)

```
# To estimate the skewness of the ratio of means for the city data.  
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)  
k3.linear(empinf(data = city, statistic = ratio))
```

linear.approx

```
linear.approx(boot.out, L = NULL, index = 1, type = NULL,  
              t0 = NULL, t = NULL, ...)
```

boot.out	"boot"boot
L	Lempinf
index	boot.out\$statistic
type	Lempinf
t0	tLboot.out\$t0[index]t0tLt0boot.out\$t0[index]
t	t0tL
...	boot.out\$statisticLempinf


```
ft0 + sum(L * f)/nboot.array
```

```
boot.out$R
```

bootempinfcontrol

```
# Using the city data let us look at the linear approximation to the  
# ratio statistic and its logarithm. We compare these with the  
# corresponding plots for the bigcity data
```

```
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)  
city.boot <- boot(city, ratio, R = 499, stype = "w")  
bigcity.boot <- boot(bigcity, ratio, R = 499, stype = "w")  
op <- par(pty = "s", mfrow = c(2, 2))
```

```
# The first plot is for the city data ratio statistic.  
city.lin1 <- linear.approx(city.boot)  
lim <- range(c(city.boot$t, city.lin1))  
plot(city.boot$t, city.lin1, xlim = lim, ylim = lim,  
      main = "Ratio; n=10", xlab = "t*", ylab = "tL*")  
abline(0, 1)
```

```
# Now for the log of the ratio statistic for the city data.  
city.lin2 <- linear.approx(city.boot, t0 = log(city.boot$t0),  
                          t = log(city.boot$t))  
lim <- range(c(log(city.boot$t), city.lin2))  
plot(log(city.boot$t), city.lin2, xlim = lim, ylim = lim,  
      main = "Log(Ratio); n=10", xlab = "t*", ylab = "tL*")  
abline(0, 1)
```

```
# The ratio statistic for the bigcity data.  
bigcity.lin1 <- linear.approx(bigcity.boot)  
lim <- range(c(bigcity.boot$t, bigcity.lin1))  
plot(bigcity.lin1, bigcity.boot$t, xlim = lim, ylim = lim,  
      main = "Ratio; n=49", xlab = "t*", ylab = "tL*")  
abline(0, 1)
```

```
# Finally the log of the ratio statistic for the bigcity data.  
bigcity.lin2 <- linear.approx(bigcity.boot, t0 = log(bigcity.boot$t0),  
                             t = log(bigcity.boot$t))  
lim <- range(c(log(bigcity.boot$t), bigcity.lin2))  
plot(bigcity.lin2, log(bigcity.boot$t), xlim = lim, ylim = lim,  
      main = "Log(Ratio); n=49", xlab = "t*", ylab = "tL*")  
abline(0, 1)
```

```
par(op)
```

```
lines.saddle.distn
```

```
## S3 method for class 'saddle.distn'
lines(x, dens = TRUE, h = function(u) u, J = function(u) 1,
      npts = 50, lty = 1, ...)
```

```
x          "saddle.distn" saddle.distn.object
dens       TRUEFALSE
h
J          dens=TRUEJhJ
npts
lty
...        hJ
```

```
smooth.splinedens=TRUEdens=FALSE
```

```
sad.d
```

```
saddle.distn
```

```
# In this example we show how a plot such as that in Figure 9.9 of
# Davison and Hinkley (1997) may be produced. Note the large number of
# bootstrap replicates required in this example.
```

```
expdata <- rexp(12)
vfun <- function(d, i) {
  n <- length(d)
  (n-1)/n*var(d[i])
}
exp.boot <- boot(expdata,vfun, R = 9999)
exp.L <- (expdata - mean(expdata))^2 - exp.boot$t0
exp.tL <- linear.approx(exp.boot, L = exp.L)
hist(exp.tL, nclass = 50, probability = TRUE)
exp.t0 <- c(0, sqrt(var(exp.boot$t)))
exp.sp <- saddle.distn(A = exp.L/12,wdist = "m", t0 = exp.t0)
```

```
# The saddlepoint approximation in this case is to the density of
# t-t0 and so t0 must be added for the plot.
```

```
lines(exp.sp, h = function(u, t0) u+t0, J = function(u, t0) 1,
      t0 = exp.boot$t0)
```

logit

logit(p)

p NA

pp-InfInfNANA

p

[inv.logitqlogis](#)

manaus

manaus"ts"

melanoma

melanoma

melanoma

time
status
sex
age
year
thickness
ulcer

motor

motortimecycle

motor

times
accel
strata
v v

mcycle

neuro

neuro

nitrofen

nitrofen

nitrofen

conc
brood1
brood2
brood3
total

nodal

nodal

nodal

m

r

aged 01

stage 1

grade 1

xray 1

acid

norm.ci

```
norm.ci(boot.out = NULL, conf = 0.95, index = 1, var.t0 = NULL,
        t0 = NULL, t = NULL, L = NULL, h = function(t) t,
        hdot = function(t) 1, hinv = function(t) t)
```

boot.out	boott0boot.outvar.t0t
conf	
index	boot.out\$statisticboot.outt0
var.t0	var(t)
t0	boot.out
t	var.t0boot.out\$t[,index]
L	var.t0var.t0boot.outht0
h	h(t)hinvh
hdot	hhh(t0)
hinv	hhh(t)hhinv

```
var.t02*qnrm((1+conf)/2)*sqrt(var.t0)boot.outt2*t0-mean(t)t0
```

```
length(conf)
```

```
boot.ci@var.t0
```

[boot.ci](#)

```
# In Example 5.1 of Davison and Hinkley (1997), normal approximation
# confidence intervals are found for the air-conditioning data.
air.mean <- mean(aircondit$hours)
air.n <- nrow(aircondit)
air.v <- air.mean^2/air.n
norm.ci(t0 = air.mean, var.t0 = air.v)
exp(norm.ci(t0 = log(air.mean), var.t0 = 1/air.n)[2:3])

# Now a more complicated example - the ratio estimate for the city data.
ratio <- function(d, w)
  sum(d$x * w)/sum(d$u *w)
city.v <- var.linear(empinf(data = city, statistic = ratio))
norm.ci(t0 = ratio(city,rep(0.1,10)), var.t0 = city.v)
```

nuclear

nuclear

nuclear

cost
date
t1
t2
cap
pr 1
ne 1
ct 1
bw 1
cum.n
pt 1

paulsen

paulsen

paulsen

y

plot.boot

```
## S3 method for class 'boot'  
plot(x, index = 1, t0 = NULL, t = NULL, jack = FALSE,  
      qdist = "norm", nclass = NULL, df, ...)
```

x	"boot"
index	boot.out\$t0
t0	boot.out\$t0[index]tNULL
t	boot.out\$t[,index]boot.out\$t
jack	
qdist	"norm""chisq"
nclass	ceiling(length(t)/25)
df	qdist"chisq"
...	jackTRUEjack.after.bootjack.after.boot


```

t0t0tt0histnclass
mean(t)sqrt(var(t))
jackTRUEjack.after.boot

```

```

boot.out

```

```

bootjack.after.bootprint.boot

```

```

# We fit an exponential model to the air-conditioning data and use
# that for a parametric bootstrap. Then we look at plots of the
# resampled means.
air.rg <- function(data, mle) rexp(length(data), 1/mle)

air.boot <- boot(aircondit$hours, mean, R = 999, sim = "parametric",
               ran.gen = air.rg, mle = mean(aircondit$hours))
plot(air.boot)

# In the difference of means example for the last two series of the
# gravity data
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
grav.fun <- function(dat, w) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2] - mns[1], s2hat)
}

grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w", strata = grav1[, 2])
plot(grav.boot)
# now suppose we want to look at the studentized differences.
grav.z <- (grav.boot$t[, 1]-grav.boot$t0[1])/sqrt(grav.boot$t[, 2])
plot(grav.boot, t = grav.z, t0 = 0)

# In this example we look at the one of the partial correlations for the
# head dimensions in the dataset frets.
frets.fun <- function(data, i) {
  pcorr <- function(x) {
    # Function to find the correlations and partial correlations between
    # the four measurements.
    v <- cor(x)
    v.d <- diag(var(x))
    iv <- solve(v)
  }
}

```

```

        iv.d <- sqrt(diag(iv))
        iv <- - diag(1/iv.d) %*% iv %*% diag(1/iv.d)
        q <- NULL
        n <- nrow(v)
        for (i in 1:(n-1))
            q <- rbind( q, c(v[i, 1:i], iv[i,(i+1):n]) )
        q <- rbind( q, v[n, ] )
        diag(q) <- round(diag(q))
        q
    }
    d <- data[i, ]
    v <- pcorr(d)
    c(v[1,], v[2,], v[3,], v[4,])
}
frets.boot <- boot(log(as.matrix(frets)), frets.fun, R = 999)
plot(frets.boot, index = 7, jack = TRUE, stinf = FALSE, useJ = FALSE)

```

poisons

poisons

poisons

time

poison 123

treat ABCD

polar

polar

polar

lat
long

print.boot

```
print()"boot"bootcensboottilt.boottsboot
```

```
## S3 method for class 'boot'  
print(x, digits = getOption("digits"),  
      index = 1:ncol(boot.out$t), ...)
```

```
x           "boot"  
digits  
index  
...
```

```
boot.out$t0tsbootorig.t = FALSE
```

```
bootcensbootimp.momentsplot.boottilt.boottsboot
```

```
print.bootci
```

```
print()"bootci"
```

```
## S3 method for class 'bootci'  
print(x, hinv = NULL, ...)
```

```
x          boot.ci  
hinv  
...
```

```
boot.ciboot.ci  
Warning : Intervals used Extreme Quantiles
```

```
Some intervals may be unstable
```

```
ci.out
```

```
boot.ci
```

```
print.saddle.distn
```

```
print()"saddle.distn"
```

```
## S3 method for class 'saddle.distn'  
print(x, ...)
```

```
x          "saddle.distn"saddle.distn  
...
```

```
lines.saddle.distnsaddle.distn
```

`print.simplex`

```
print()"simplex"
```

```
## S3 method for class 'simplex'  
print(x, ...)
```

```
x          "simplex"simplex  
...
```

```
x
```

```
simplex
```

`remission`

```
remission
```

```
remission
```

```
LI
```

```
m
```

```
r m
```

saddle

u

```
saddle(A = NULL, u = NULL, wdist = "m", type = "simp", d = NULL,
       d1 = 1, init = rep(0.1, d), mu = rep(0.5, n), LR = FALSE,
       strata = NULL, K.adj = NULL, K2 = NULL)
```

A	K.adjK2
u	K.adjK2
wdist	"m""p""b""o"K.adjK2wdist"o"
type	"simp""cond"wdist"o""m"type"simp"
d	wdist = "o"ncol(A)
d1	type"cond"length(u)d1d1
init	wdist"m""o"nlmin
mu	wdist"m""p""b""munrow(A)mu
LR	TRUE
strata	
K.adj	wdist"o"zetaK(zeta)-u%%zetaK(zeta)
K2	zetaK(zeta)wdist"o"K.adjK.adj

```
wdist"o""m"nlminK.adjzetaglm(A)%%y = ut()simplexglmA
```

```
spa
zeta.hat
zeta2.hat      type"cond"type
```

[saddle.distnsimplex](#)

```

# To evaluate the bootstrap distribution of the mean failure time of
# air-conditioning equipment at 80 hours
saddle(A = aircondit$hours/12, u = 80)

# Alternatively this can be done using a conditional poisson
saddle(A = cbind(aircondit$hours/12,1), u = c(80, 12),
       wdlist = "p", type = "cond")

# To use the Lugananni-Rice approximation to this
saddle(A = cbind(aircondit$hours/12,1), u = c(80, 12),
       wdlist = "p", type = "cond",
       LR = TRUE)

# Example 9.16 of Davison and Hinkley (1997) calculates saddlepoint
# approximations to the distribution of the ratio statistic for the
# city data. Since the statistic is not in itself a linear combination
# of random Variables, its distribution cannot be found directly.
# Instead the statistic is expressed as the solution to a linear
# estimating equation and hence its distribution can be found. We
# get the saddlepoint approximation to the pdf and cdf evaluated at
# t = 1.25 as follows.
jacobian <- function(dat,t,zeta)
{
  p <- exp(zeta*(dat$x-t*dat$u))
  abs(sum(dat$u*p)/sum(p))
}
city.sp1 <- saddle(A = city$x-1.25*city$u, u = 0)
city.sp1$spa[1] <- jacobian(city, 1.25, city.sp1$zeta.hat) * city.sp1$spa[1]
city.sp1

```

saddle.distn

control

```

saddle.distn(A, u = NULL, alpha = NULL, wdlist = "m",
             type = "simp", npts = 20, t = NULL, t0 = NULL,
             init = rep(0.1, d), mu = rep(0.5, n), LR = FALSE,
             strata = NULL, ...)

```

A	t
u	AuAAwdlist = "cond"uncol(A)uAu
alpha	
wdlist	"m""p""b"
type	"simp""cond"wdlist"m"type"simpl
npts	
t	ttnptslength(t)t

```

t0          tt0t0t0[1]
init        wdist"m"nlmin
mu
LR          LRTRUE
strata      wdist"m"
...         Au

```

```

alphat0[1]-2*t0[2]t0[1]-4*t0[2]t0[1]-8*t0[2]min(alpha)/10min(alpha)/1000
min(alpha)/10t0[1]npts/2npts/2t0[1]
"Unable to find range"t0[2]t0[2]

```

```

"saddle.distn"saddle.distn.object

```

```

lines.saddle.distnsaddlesaddle.distn.objectsmooth.spline

```

```

# The bootstrap distribution of the mean of the air-conditioning
# failure data: fails to find value on R (and probably on S too)
air.t0 <- c(mean(aircondit$hours), sqrt(var(aircondit$hours)/12))
## Not run: saddle.distn(A = aircondit$hours/12, t0 = air.t0)

# alternatively using the conditional poisson
saddle.distn(A = cbind(aircondit$hours/12, 1), u = 12, wdists = "p",
             type = "cond", t0 = air.t0)

# Distribution of the ratio of a sample of size 10 from the bigcity
# data, taken from Example 9.16 of Davison and Hinkley (1997).
ratio <- function(d, w) sum(d$x *w)/sum(d$u * w)
city.v <- var.linear(empinf(data = city, statistic = ratio))
bigcity.t0 <- c(mean(bigcity$x)/mean(bigcity$u), sqrt(city.v))
Afn <- function(t, data) cbind(data$x - t*data$u, 1)
ufn <- function(t, data) c(0,10)
saddle.distn(A = Afn, u = ufn, wdists = "b", type = "cond",
             t0 = bigcity.t0, data = bigcity)

# From Example 9.16 of Davison and Hinkley (1997) again, we find the
# conditional distribution of the ratio given the sum of city$u.
Afn <- function(t, data) cbind(data$x-t*data$u, data$u, 1)
ufn <- function(t, data) c(0, sum(data$u), 10)
city.t0 <- c(mean(city$x)/mean(city$u), sqrt(city.v))
saddle.distn(A = Afn, u = ufn, wdists = "p", type = "cond", t0 = city.t0,
             data = city)

```

saddle.distn.object

saddle.distn

saddle.distn

"saddle.distn"linesprint

"saddle.distn"

alpha

ttype"simp"type"cond"2*d-1dAA(t,...{ })d

smooth.splinesmooth.spline

saddle.distn

lines.saddle.distnsaddle.distnprint.saddle.distn

salinity

salinity

salinity

sal

lag lagsal

trend

dis sal

simplex

$a\%x A1\%x \leq b1 A2\%x \geq b2 A3\%x = b3x \geq 0$

```
simplex(a, A1 = NULL, b1 = NULL, A2 = NULL, b2 = NULL, A3 = NULL,  
       b3 = NULL, maxi = FALSE, n.iter = n + 2 * m, eps = 1e-10)
```

a	n
A1	$m1n \leq$
b1	$m1 \leq A1b1$
A2	$m2n \geq$
b2	$m2 \geq A2b2x \geq 0$
A3	$m3n$
b3	$m3A3b3$
maxi	FALSEmaxiTRUE
n.iter	$n+2*(m1+m2+m3)$
eps	

$\geq \leq$

"simplex"[simplex.object](#)

$x[i] \geq b2[i]x[i] = x[i]-b2[i]$

```
# This example is taken from Exercise 7.5 of Gill, Murray and Wright (1991).  
enj <- c(200, 6000, 3000, -200)  
fat <- c(800, 6000, 1000, 400)  
vitx <- c(50, 3, 150, 100)  
vity <- c(10, 10, 75, 100)  
vitz <- c(150, 35, 75, 5)  
simplex(a = enj, A1 = fat, b1 = 13800, A2 = rbind(vitx, vity, vitz),  
       b2 = c(600, 300, 550), maxi = TRUE)
```

simplex.object

simplex

simplex

"saddle.distn"print

"simplex"

x

-101

soln

NULL

NULL

m1+m2+m3n+m1+m2m1+m2m1+m2+m3

n+1n+m1n+m1+1n+m2n+m2n+m2solved-1

m1A1%*%x + slack = b1

m2A2%*%x - surplus = b2

m1+m2+m3

print.simplexsimplex

smooth.f

thetatheta

smooth.f(theta, boot.out, index = 1, t = boot.out\$t[, index],
width = 0.5)

theta	thetatheta
boot.out	boot
index	boot.out\$statistictindex
t	boot.out\$Rboot.out
width	width*sswidthwidth

```
ttheta
```

```
length(theta)boot.out$data[[theta]]length(theta)length(theta)theta
```

```
bootexp.tiltboot
```

```
# Example 9.8 of Davison and Hinkley (1997) requires tilting the resampling
# distribution of the studentized statistic to be centred at the observed
# value of the test statistic 1.84. In the book exponential tilting was used
# but it is also possible to use smooth.f.
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2] - mns[1], s2hat, (mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w",
  strata = grav1[, 2], orig = grav.z0[1])
grav.sm <- smooth.f(grav.z0[3], grav.boot, index = 3)

# Now we can run another bootstrap using these weights
grav.boot2 <- boot(grav1, grav.fun, R = 499, stype = "w",
  strata = grav1[, 2], orig = grav.z0[1],
  weights = grav.sm)

# Estimated p-values can be found from these as follows
mean(grav.boot$t[, 3] >= grav.z0[3])
imp.prob(grav.boot2, t0 = -grav.z0[3], t = -grav.boot2$t[, 3])

# Note that for the importance sampling probability we must
# multiply everything by -1 to ensure that we find the correct
# probability. Raw resampling is not reliable for probabilities
# greater than 0.5. Thus
1 - imp.prob(grav.boot2, index = 3, t0 = grav.z0[3])$raw
# can give very strange results (negative probabilities).
```

sunspot

sunspot

survival

survival

survival

dose

surv

tau
tau
tau
rate decay 1rhopienu
tilt.boot

```
tilt.boot(data, statistic, R, sim = "ordinary", stype = "i",
          strata = rep(1, n), L = NULL, theta = NULL,
          alpha = c(0.025, 0.975), tilt = TRUE, width = 0.5,
          index = 1, ...)
```

data	
statistic	datatilt.boot
R	RRlength(R)1+length(theta)
sim	"ordinary""balanced"bootboot
stype	statisticstype"i""w""f"
strata	
L	tiltTRUEtiltTRUEtilt.bootempinf

```

theta      thetaR[1]thetathetaalpha
alpha      R[1]thetathetaR[1]min(c(alpha, 1-alpha))*R[1] > 5theta
tilt       TRUEsmooth.ftiltFALSER[1]R[1]
width      tiltFALSEsmooth.fsmooth.f
index      statisticstatistic
...        statisticstatistic

"boot"

t0
t          sum(R)R[1]
R
data
statistic  statistic
sim        "ordinary""balanced"
stype      stype
call       tilt.boot
strata
weights    R[1]R[1]
theta      thetaalpha

```

[bootexp.tiltImp.Estimatesimp.weightssmooth.f](#)

```

# Note that these examples can take a while to run.

# Example 9.9 of Davison and Hinkley (1997).
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
  strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
  d <- dat[, 1]
  ns <- tabulate(strata)
  w <- w/tapply(w, strata, sum)[strata]
  mns <- as.vector(tapply(d * w, strata, sum)) # drop names
  mn2 <- tapply(d * d * w, strata, sum)
  s2hat <- sum((mn2 - mns^2)/ns)
  c(mns[2]-mns[1],s2hat,(mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
tilt.boot(grav1, grav.fun, R = c(249, 375, 375), stype = "w",
  strata = grav1[,2], tilt = TRUE, index = 3, orig = grav.z0[1])

```

```

# Example 9.10 of Davison and Hinkley (1997) requires a balanced
# importance resampling bootstrap to be run. In this example we
# show how this might be run.
acme.fun <- function(data, i, bhat) {
  d <- data[i,]
  n <- nrow(d)
  d.lm <- glm(d$acme~d$market)
  beta.b <- coef(d.lm)[2]
  d.diag <- boot::glm.diag(d.lm)
  SSx <- (n-1)*var(d$market)
  tmp <- (d$market-mean(d$market))*d.diag$res*d.diag$sd
  sr <- sqrt(sum(tmp^2))/SSx
  c(beta.b, sr, (beta.b-bhat)/sr)
}
acme.b <- acme.fun(acme, 1:nrow(acme), 0)
acme.boot1 <- tilt.boot(acme, acme.fun, R = c(499, 250, 250),
  stype = "i", sim = "balanced", alpha = c(0.05, 0.95),
  tilt = TRUE, index = 3, bhat = acme.b[1])

```

tsboot

R

```

tsboot(tseries, statistic, R, l = NULL, sim = "model",
  endcorr = TRUE, n.sim = NROW(tseries), orig.t = TRUE,
  ran.gen, ran.args = NULL, norm = TRUE, ...,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("boot.ncpus", 1L), cl = NULL)

```

```

tseries
statistic      tseriesstatisticn.simtseriesstatisticstboot
R
sim            "model""fixed"l"geom"l"scramble"
l              sim"fixed"lsim"geom"lltseriesisim"model"
endcorr        sim"fixed"sim"geom"endcorrTRUEendcorrsim"model""scramble"
n.sim          n.simtseriesn.sim
orig.t         statistictseriesstatisticseriesstatisticseriesorig.tFALSE
ran.gen        sim"model"tseriesn.simtseriesseriesn.simran.genn.simran.args
               ran.gensim"model"ran.genarima.simran.gen
ran.args       ran.genran.genran.argsran.argsran.argsNULLran.genran.gen
norm           normFALSE
...            statistictboot
parallelncpuscl
               boot

```



```
sim"fixed"ln.simsim"geom"lran.gentsboottseries
```

```
bootsim = "parametric"
```

```
"boot"
```

```
t0      orig.tTRUEt0statistic(tseries,...{})NULL
t        statistic
R        Rtsboot
tseries
statistic statistic
sim
endcorr  endcorrsim"fixed"TRUEsim"geom"
n.sim    n.sim
l        lNULL
ran.gen  ran.gen
ran.args ran.gen
call     tsboot
```

```
bootarima.sim
```

```
lynx.fun <- function(tsb) {
  ar.fit <- ar(tsb, order.max = 25)
  c(ar.fit$order, mean(tsb), tsb)
}

# the stationary bootstrap with mean block length 20
lynx.1 <- tsboot(log(lynx), lynx.fun, R = 99, l = 20, sim = "geom")

# the fixed block bootstrap with length 20
lynx.2 <- tsboot(log(lynx), lynx.fun, R = 99, l = 20, sim = "fixed")

# Now for model based resampling we need the original model
# Note that for all of the bootstraps which use the residuals as their
# data, we set orig.t to FALSE since the function applied to the residual
# time series will be meaningless.
lynx.ar <- ar(log(lynx))
lynx.model <- list(order = c(lynx.ar$order, 0, 0), ar = lynx.ar$ar)
lynx.res <- lynx.ar$resid[!is.na(lynx.ar$resid)]
```

```

lynx.res <- lynx.res - mean(lynx.res)

lynx.sim <- function(res,n.sim, ran.args) {
  # random generation of replicate series using arima.sim
  rg1 <- function(n, res) sample(res, n, replace = TRUE)
  ts.orig <- ran.args$ts
  ts.mod <- ran.args$model
  mean(ts.orig)+ts(arima.sim(model = ts.mod, n = n.sim,
                             rand.gen = rg1, res = as.vector(res)))
}

lynx.3 <- tsboot(lynx.res, lynx.fun, R = 99, sim = "model", n.sim = 114,
                 orig.t = FALSE, ran.gen = lynx.sim,
                 ran.args = list(ts = log(lynx), model = lynx.model))

# For "post-blackening" we need to define another function
lynx.black <- function(res, n.sim, ran.args) {
  ts.orig <- ran.args$ts
  ts.mod <- ran.args$model
  mean(ts.orig) + ts(arima.sim(model = ts.mod,n = n.sim,innov = res))
}

# Now we can run apply the two types of block resampling again but this
# time applying post-blackening.
lynx.1b <- tsboot(lynx.res, lynx.fun, R = 99, l = 20, sim = "fixed",
                  n.sim = 114, orig.t = FALSE, ran.gen = lynx.black,
                  ran.args = list(ts = log(lynx), model = lynx.model))

lynx.2b <- tsboot(lynx.res, lynx.fun, R = 99, l = 20, sim = "geom",
                  n.sim = 114, orig.t = FALSE, ran.gen = lynx.black,
                  ran.args = list(ts = log(lynx), model = lynx.model))

# To compare the observed order of the bootstrap replicates we
# proceed as follows.
table(lynx.1$t[, 1])
table(lynx.1b$t[, 1])
table(lynx.2$t[, 1])
table(lynx.2b$t[, 1])
table(lynx.3$t[, 1])
# Notice that the post-blackened and model-based bootstraps preserve
# the true order of the model (11) in many more cases than the others.

```

tuna

tuna

tuna

y

urine

urine

urine

r

gravity

ph

osmo

cond

urea

calc

`var.linear`

```
var.linear(L, strata = NULL)
```

```
L          empinf  
strata
```

```
L
```

```
empinflinear.approxk3.linear
```

```
# To estimate the variance of the ratio of means for the city data.  
ratio <- function(d,w) sum(d$x * w)/sum(d$u * w)  
var.linear(empinf(data = city, statistic = ratio))
```

`wool`

```
wool"ts"
```


class

batchSOM

```
batchSOM(data, grid = somgrid(), radii, init)
```

```
data
grid      somgrid
radii     radii
init      data
```

```
"SOM"
grid      "somgrid"
codes
```

somgridSOM

```

require(graphics)
data(crabs, package = "MASS")

lcrabs <- log(crabs[, 4:8])
crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, rep(50,4))])
gr <- somgrid(topo = "hexagonal")
crabs.som <- batchSOM(lcrabs, gr, c(4, 4, 2, 2, 1, 1, 1, 0, 0))
plot(crabs.som)

bins <- as.numeric(knn1(crabs.som$codes, lcrabs, 0:47))
plot(crabs.som$grid, type = "n")
symbols(crabs.som$grid$pts[, 1], crabs.som$grid$pts[, 2],
        circles = rep(0.4, 48), inches = FALSE, add = TRUE)
text(crabs.som$grid$pts[bins, ] + rnorm(400, 0, 0.1),
     as.character(crabs.grp))

```

condense

```
condense(train, class, store, trace = TRUE)
```

```

train
class
store
trace

```

[reduce.nnmultiedit](#)

```

train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
keep <- condense(train, cl)
knn(train[keep, , drop=FALSE], test, cl[keep])
keep2 <- reduce.nn(train, keep, cl)
knn(train[keep2, , drop=FALSE], test, cl[keep2])

```

knn

kk

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

train

test

cl

k

l doubtk-lk

prob prob

use.all kkk

doubtNA

[knn1knn.cv](#)

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn(train, test, cl, k = 3, prob=TRUE)
attributes(.Last.value)
```

knn.cv

```
knn.cv(train, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```



```
train
cl
k
l          doubtk-lk
prob      prob
use.all   kkk
```

```
trainkk
```

```
doubtNA
```

knn

```
train <- rbind(iris3[,1], iris3[,2], iris3[,3])
cl <- factor(c(rep("s",50), rep("c",50), rep("v",50)))
knn.cv(train, cl, k = 3, prob = TRUE)
attributes(.Last.value)
```

```
knn1
```

```
knn1(train, test, cl)
```

```
train
test
cl
```

knn

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn1(train, test, cl)
```

lvq1

```
lvq1(x, cl, codebk, niter = 100 * nrow(codebk$x), alpha = 0.03)
```

```
x
cl
codebk
niter
alpha
```

```
niter
```

```
xcl
```

lvqinitolvq1lvq2lvq3lvqtest

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd0 <- olvq1(train, cl, cd)
lvqtest(cd0, train)
cd1 <- lvq1(train, cl, cd0)
lvqtest(cd1, train)
```

lvq2

```
lvq2(x, cl, codebk, niter = 100 * nrow(codebk$x), alpha = 0.03,  
     win = 0.3)
```

```
x  
cl  
codebk  
niter  
alpha  
win
```

```
niter
```

```
xcl
```

[lvqinitlvq1olvg1lvq3lvqtest](#)

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])  
test  <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])  
cl    <- factor(c(rep("s",25), rep("c",25), rep("v",25)))  
cd    <- lvqinit(train, cl, 10)  
lvqtest(cd, train)  
cd0   <- olvg1(train, cl, cd)  
lvqtest(cd0, train)  
cd2   <- lvq2(train, cl, cd0)  
lvqtest(cd2, train)
```

lvq3

```
lvq3(x, cl, codebk, niter = 100*nrow(codebk$x), alpha = 0.03,  
     win = 0.3, epsilon = 0.1)
```

```
x  
cl  
codebk  
niter  
alpha  
win  
epsilon
```

```
niter
```

```
xcl
```

[lvqinitlvq1olvg1lvq2lvqtest](#)

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])  
test  <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])  
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))  
cd <- lvqinit(train, cl, 10)  
lvqtest(cd, train)  
cd0 <- olvg1(train, cl, cd)  
lvqtest(cd0, train)  
cd3 <- lvq3(train, cl, cd0)  
lvqtest(cd3, train)
```

lvqinit

```
lvqinit(x, cl, size, prior, k = 5)
```

```
x          np
cl          n
size        min(round(0.4*ng*(ng-1 + p/2),0), n)ng
prior
k
```

```
size
```

```
xcl
```

[lvq1lvq2lvq3ol](#)[lvq1lvqtest](#)

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test  <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl    <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd    <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd1   <- olvq1(train, cl, cd)
lvqtest(cd1, train)
```

lvqtest

lvqtest(codebk, test)

codebk

test

x

[lvqinitolvq1](#)

The function is currently defined as
function(codebk, test) knn1(codebk\$x, test, codebk\$c1)

multiedit

multiedit(x, class, k = 1, V = 3, I = 5, trace = TRUE)

x

class

k

V

I

trace

[condensereduce.nn](#)

```
tr <- sample(1:50, 25)
train <- rbind(iris3[tr,,1], iris3[tr,,2], iris3[tr,,3])
test <- rbind(iris3[-tr,,1], iris3[-tr,,2], iris3[-tr,,3])
cl <- factor(c(rep(1,25),rep(2,25), rep(3,25)), labels=c("s", "c", "v"))
table(cl, knn(train, test, cl, 3))
ind1 <- multiedit(train, cl, 3)
length(ind1)
table(cl, knn(train[ind1, , drop=FALSE], test, cl[ind1], 1))
ntrain <- train[ind1,]; ncl <- cl[ind1]
ind2 <- condense(ntrain, ncl)
length(ind2)
table(cl, knn(ntrain[ind2, , drop=FALSE], test, ncl[ind2], 1))
```

olvq1

```
olvq1(x, cl, codebk, niter = 40 * nrow(codebk$x), alpha = 0.3)
```

```
x
cl
codebk
niter
alpha
```

```
niter
```

```
xcl
```

lvqinitlvqtestlvq1lvq2lvq3

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test  <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl    <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
cd    <- lvqinit(train, cl, 10)
lvqtest(cd, train)
cd1   <- olvq1(train, cl, cd)
lvqtest(cd1, train)
```

reduce.nn

condense

reduce.nn(train, ind, class)

train

ind

condense

class

condensemultiedit

```
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test  <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl    <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
keep  <- condense(train, cl)
knn(train[keep,], test, cl[keep])
keep2 <- reduce.nn(train, keep, cl)
knn(train[keep2,], test, cl[keep2])
```

SOM

```
SOM(data, grid = somgrid(), rlen = 10000, alpha, radii, init)
```

```
data
grid      somgrid
rlen      alphasradii
alpha     alphasrlen
radii     alphasrlen
init      data
```

```
alpharadii
```

```
"SOM"
```

```
grid      "somgrid"
codes
```

[somgridbatchSOM](#)

```
require(graphics)
data(crabs, package = "MASS")

lcrabs <- log(crabs[, 4:8])
crabs.grp <- factor(c("B", "b", "O", "o")[rep(1:4, rep(50,4))])
gr <- somgrid(topo = "hexagonal")
crabs.som <- SOM(lcrabs, gr)
plot(crabs.som)

## 2-phase training
crabs.som2 <- SOM(lcrabs, gr,
  alpha = list(seq(0.05, 0, length.out = 1e4), seq(0.02, 0, length.out = 1e5)),
  radii = list(seq(8, 1, length.out = 1e4), seq(4, 1, length.out = 1e5)))
plot(crabs.som2)
```

somgrid

```
somgrid(xdim = 8, ydim = 6, topo = c("rectangular", "hexagonal"))
```

```
## S3 method for class 'somgrid'  
plot(x, type = "p", ...)
```

```
## S3 method for class 'SOM'  
plot(x, ...)
```

```
xdim ydim  
topo  
x          "somgrid" "SOM"  
type...
```

```
"somgrid"  
"SOM" stars
```

```
somgrid"somgrid"  
pts  
xdim ydim topo    somgrid
```

```
batchSOM SOM
```


cluster

agnes

```
agnes(x, diss = inherits(x, "dist"), metric = "euclidean",
      stand = FALSE, method = "average", par.method,
      keep.diss = n < 100, keep.data = !diss, trace.lev = 0)
```

x	diss
	xdaisydist
diss	distdissimilarityxx
metric	"euclidean""manhattan"x
stand	xx
method	"average""single""complete""ward""weighted""flexible"par.method "gaverage""average"par.method "average"
par.method	method"flexible""gaverage""gaverage"
keep.disskeep.data	xFALSE
trace.lev	0

```
agneshclustagnesagnes.objectplot.agnes  
agnes
```

```
method="average"  
method="single"  
method="complete"
```

method = "flexible" C_1C_2Q

$D(C_1 \cup C_2, Q) = \alpha_1 * D(C_1, Q) + \alpha_2 * D(C_2, Q) + \beta * D(C_1, C_2) + \gamma * |D(C_1, Q) - D(C_2, Q)|,$

$(\alpha_1, \alpha_2, \beta, \gamma)$ par.methodpar.method= α par.method($\alpha_1 = \alpha_2 = \alpha, \beta = 1 - 2\alpha, \gamma = 0$)

length(par.method) == $3\gamma = 0$

method = "flexible"par.methodmergemethod="weighted"method="flexible",
par.method = 0.5method= "single"method="flexible", par.method = c(.5,.5,0,-.5)
method="complete"method="flexible", par.method = c(.5,.5,0,+.5)

method = "gaverage""average""flexible" $\alpha_1\alpha_2n_1n_2C_1C_2$

$$\alpha_j = \alpha'_j \frac{n_1}{n_1 + n_2},$$

$\alpha'_1\alpha'_2$ par.method($\alpha_1, \alpha_2, \beta, \gamma$)($\alpha_1, \alpha_2, \beta$) $\gamma = 0$

β par.method

$$-1 \leq \beta < 1,$$

$\beta\alpha'_1\alpha'_2$

$$\alpha'_j = 1 - \beta,$$

$\gamma = 0$

β par.methodpar.method β

method = "gaverage", par.method = 0par.method = c(1,1,0,0)agnes()"average"

"agnes""twins"[agnes.object](#)

[agnesdianamona](#)

[pamclarafanny](#)

"gaverage"

[agnes.objectdaisydianadisthclustplot.agnestwins.object](#)

```

data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)
agn1
plot(agn1)

op <- par(mfrow=c(2,2))
agn2 <- agnes(daisy(votes.repub), diss = TRUE, method = "complete")
plot(agn2)
## alpha = 0.625 ==> beta = -1/4 is "recommended" by some
agnS <- agnes(votes.repub, method = "flexible", par.method = 0.625)
plot(agnS)
par(op)

## "show" equivalence of three "flexible" special cases
d.vr <- daisy(votes.repub)
a.wgt <- agnes(d.vr, method = "weighted")
a.sing <- agnes(d.vr, method = "single")
a.comp <- agnes(d.vr, method = "complete")
iC <- -(6:7) # not using 'call' and 'method' for comparisons
stopifnot(
  all.equal(a.wgt[iC], agnes(d.vr, method="flexible", par.method = 0.5)[iC]) ,
  all.equal(a.sing[iC], agnes(d.vr, method="flex", par.method= c(.5,.5,0, -.5))[iC]),
  all.equal(a.comp[iC], agnes(d.vr, method="flex", par.method= c(.5,.5,0, +.5))[iC]))

## Exploring the dendrogram structure
(d2 <- as.dendrogram(agn2)) # two main branches
d2[[1]] # the first branch
d2[[2]] # the 2nd one { 8 + 42 = 50 }
d2[[1]][[1]]# first sub-branch of branch 1 .. and shorter form
identical(d2[[c(1,1)]],
  d2[[1]][[1]])
## a "textual picture" of the dendrogram :
str(d2)

data(agriculture)

## Plot similar to Figure 7 in ref
## Not run: plot(agnes(agriculture), ask = TRUE)

data(animals)
aa.a <- agnes(animals) # default method = "average"
aa.ga <- agnes(animals, method = "gaverage")
op <- par(mfcol=1:2, mgp=c(1.5, 0.6, 0), mar=c(.1+ c(4,3,2,1)),
  cex.main=0.8)
plot(aa.a, which.plots = 2)
plot(aa.ga, which.plots = 2)
par(op)

## Show how "gaverage" is a "generalized average":
aa.ga.0 <- agnes(animals, method = "gaverage", par.method = 0)
stopifnot(all.equal(aa.ga.0[iC], aa.a[iC]))

```

agnes.object

"agnes"

agnes

order

order.lab order

height

ac

acac

merge merge

diss "dissimilarity"[dissimilarity.object](#)

data standagnes

[agnes](#)

"agnes"[printsummaryplot](#)[as.dendrogram](#)

[cutree](#)(x, *)

"agnes""twins"[pltree](#)[as.hclust](#)[agnes](#)[as.hclust](#)()

[agnes](#)[diana](#)[as.hclust](#)[hclustplot.agnes](#)[twins.object](#)

[cutree](#)

```
data(agriculture)
ag.ag <- agnes(agriculture)
class(ag.ag)
pltree(ag.ag) # the dendrogram
```

```
## cut the dendrogram -> get cluster assignments:
(ck3 <- cutree(ag.ag, k = 3))
(ch6 <- cutree(as.hclust(ag.ag), h = 6))
stopifnot(identical(unname(ch6), ck3))
```

agriculture

```
data(agriculture)
```

```
x  
y
```

agnes

agnesdaisydiana

```
data(agriculture)
```

```
## Compute the dissimilarities using Euclidean metric and without  
## standardization
```

```
daisy(agriculture, metric = "euclidean", stand = FALSE)
```

```
## 2nd plot is similar to Figure 3 in Struyf et al (1996)  
plot(pam(agriculture, 2))
```

```
## Plot similar to Figure 7 in Struyf et al (1996)  
## Not run: plot(agnes(agriculture), ask = TRUE)
```

```
## Plot similar to Figure 8 in Struyf et al (1996)  
## Not run: plot(diana(agriculture), ask = TRUE)
```

animals

```
data(animals)
```

agnes

```
data(animals)
apply(animals,2, table) # simple overview

ma <- mona(animals)
ma
## Plot similar to Figure 10 in Struyf et al (1996)
plot(ma)
```

bannerplot

barplot

```
bannerplot(x, w = rev(x$height), fromLeft = TRUE,
           main=NULL, sub=NULL, xlab = "Height", adj = 0,
           col = c(2, 0), border = 0, axes = TRUE, frame.plot = axes,
           rev.xax = !fromLeft, xax.pretty = TRUE,
           labels = NULL, nmax.lab = 35, max.strlen = 5,
           yax.do = axes && length(x$order) <= nmax.lab,
           yaxRight = fromLeft, y.mar = 2.4 + max.strlen/2.5, ...)
```

x	orderorder.labheightw
w	
fromLeft	
mainsub	title
xlab	plot.agnes
adj	title (main,sub)
col	
border	
axes	
frame.plot	border = 0
rev.xax	plot.diana
xax.pretty	pretty ()xax.pretty = FALSE
labels	x
nmax.lab	
max.strlen	
yax.do	
yaxRight	
y.mar	
...	par

[plot.agnes](#)[plot.diana](#)[plot.mona](#)

```
data(agriculture)
bannerplot(agnes(agriculture), main = "Bannerplot")
```

chorSub

chorizon

data(chorSub)

chorizon

```
data(chorizon, package = "mvoutlier")
chorSub <- round(100*scale(chorizon[,101:110]))[190:250,]
storage.mode(chorSub) <- "integer"
colnames(chorSub) <- gsub("_.*", '', colnames(chorSub))
```

chorizon

```
data(chorSub)
summary(chorSub)
pairs(chorSub, gap= .1)# some outliers
```

clara

"clara"[listk](#)

```
clara(x, k, metric = c("euclidean", "manhattan", "jaccard"),
      stand = FALSE, cluster.only = FALSE, samples = 5,
      sampsize = min(n, 40 + 2 * k), trace = 0, medoids.x = TRUE,
      keep.data = medoids.x, rngR = FALSE, pamLike = FALSE, correct.d = TRUE)
```

```

x
k          0 < k < nrow(x)
metric

stand      x
cluster.only
samples    N = 5samples
sampsizes  jsamplesizekn = nrow(x)j^2j = samplesize
trace
medoids.x  xFALSEkeep.datai.medx
keep.data  standFALSEclusplot()medoids.x = FALSE
rngR       clara()
pamLike    pampam()pamLike = FALSE
correct.d  NAx
           correct.d

```

```

clarapamsamplesize

```

```

k
pam

```

```

sampsizes
cluster.only
clara(x,k, cluster.only=TRUE)
clara(x,k)$clustering

```

```

cluster.only"clara"clara.object
cluster.onlyn1:k

```

```

2^16 = 65536rngR = TRUEclara()set.seed()clara
clarakO(n × p) + O(j^2)j = samplesize(n,p) = dim(x)kO(n × p × j^2 × N)N = samples
pampam

```

```

agnes"jaccard"@ownedoutcomes.comtrace

```

```

agnesclara.objectpampartition.objectplot.partition

```

```

## generate 500 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(200,0,8), rnorm(200,0,8)),
           cbind(rnorm(300,50,8), rnorm(300,50,8)))
clarax <- clara(x, 2, samples=50)
clarax
clarax$clusinfo
## using pamLike=TRUE gives the same (apart from the 'call'):
all.equal(clarax[-8],
          clara(x, 2, samples=50, pamLike = TRUE)[-8])
plot(clarax)

## cluster.only = TRUE -- save some memory/time :
clclus <- clara(x, 2, samples=50, cluster.only = TRUE)
stopifnot(identical(clclus, clarax$clustering))

## 'xclara' is an artificial data set with 3 clusters of 1000 bivariate
## objects each.
data(xclara)
(clx3 <- clara(xclara, 3))
## "better" number of samples
cl.3 <- clara(xclara, 3, samples=100)
## but that did not change the result here:
stopifnot(cl.3$clustering == clx3$clustering)
## Plot similar to Figure 5 in Struyf et al (1996)
## Not run: plot(clx3, ask = TRUE)

## Try 100 times *different* random samples -- for reliability:
nSim <- 100
nCl <- 3 # = no.classes
set.seed(421)# (reproducibility)
cl <- matrix(NA,nrow(xclara), nSim)
for(i in 1:nSim)
  cl[,i] <- clara(xclara, nCl, medoids.x = FALSE, rngR = TRUE)$clustering
tcl <- apply(cl,1, tabulate, nbins = nCl)
## those that are not always in same cluster (5 out of 3000 for this seed):
(iDoubt <- which(apply(tcl,2, function(n) all(n < nSim))))
if(length(iDoubt)) { # (not for all seeds)
  tabD <- tcl[,iDoubt, drop=FALSE]
  dimnames(tabD) <- list(cluster = paste(1:nCl), obs = format(iDoubt))
  t(tabD) # how many times in which clusters
}

```

clara.object

"clara"[clara](#)

clara

sample clara

```

medoids      NULL clara(*, medoids.x=FALSE)i.med
i.med        medoids medoids <- x[i.med,]x clara(x,*)
clustering   partition.object
objective
clusinfo
diss         partition.object
silinfo      partition.object
call         partition.object
data         partition.object

```

```

"clara"printsummary
"clara""partition"plotclusplotclara

```

```

claradissimilarity.objectpartition.objectplot.partition

```

```
clusGap
```

```

clusGap(k)log(W(k))E*[log(W(k))]H_0xsvd spaceH_0 = "scaledPCA"
maxSE(f, SE.f)f*SE*"firstSEmax"kf(k)"Tibs2001SEmax"

```

```

clusGap(x, FUNcluster, K.max, B = 100, d.power = 1,
        spaceH_0 = c("scaledPCA", "original"),
        verbose = interactive(), ...)

```

```

maxSE(f, SE.f,
      method = c("firstSEmax", "Tibs2001SEmax", "globalSEmax",
                  "firstmax", "globalmax"),
      SE.factor = 1)

```

```

## S3 method for class 'clusGap'
print(x, method = "firstSEmax", SE.factor = 1, ...)

```

```

## S3 method for class 'clusGap'
plot(x, type = "b", xlab = "k", ylab = expression(Gap[k]),
     main = NULL, do.arrows = TRUE,
     arrowArgs = list(col="red3", length=1/16, angle=90, code=3), ...)

```

```

x                data.frame
FUNcluster       function x, k ≥ 2 list cluster n = nrow(x) 1:kn
K.max            K
B
d.power          pdist W(k) d.power = 1 d.power = 2
spaceH0          character H0 "scaledPCA" "original" "original"
verbose
...              clusGap() FUNcluster() kmeans
f                Kf
SE.f             Kf
method           k̂k f_k
                "globalmax" which.max(f)
                "firstmax"
                "Tibs2001SEmax" k f(k) ≥ f(k + 1) − s_{k+1} k = 1 f(k + 1) − f(k)
                "firstSEmax" f() SE.factor * SE.f[] SE.factor
                clusGap() "globalSEmax" "Tibs2001SEmax"
                "globalSEmax" f() SE.factor * SE.f[] SE.factor

SE.factor        method "SE" SE.factor f
typexlabylabmain
                plot.default()
do.arrows        arrows()
arrowArgs        arrows() anglecode

<res>$Tab[, "gap"] set.seed() B = 500

clusGap(..) "clusGap"

Tab              K.max gap = E.log W - log W SE.sim gap SE.sim[k] = s_k s_k :=
                √(1 + 1/B) sd*(gap_j) sd*()

call             clusGap(..) call
spaceH0          spaceH0 match.arg()
n                nrow(x)
B                B
FUNcluster       FUNcluster

gapgapStat() gap()

spaceH0 = "original"

```

<https://bioconductor.org/packages/3.12/bioc/html/SAGx.html>

`silhouette`

`cluster.stats()`

```
### --- maxSE() methods -----
(mets <- eval(formals(maxSE)$method))
fk <- c(2,3,5,4,7,8,5,4)
sk <- c(1,1,2,1,1,3,1,1)/2
## use plot.clusGap():
plot(structure(class="clusGap", list(Tab = cbind(gap=fk, SE.sim=sk))))
## Note that 'firstmax' and 'globalmax' are always at 3 and 6 :
sapply(c(1/4, 1,2,4), function(SEf)
      sapply(mets, function(M) maxSE(fk, sk, method = M, SE.factor = SEf)))

### --- clusGap() -----
## ridiculously nicely separated clusters in 3 D :
x <- rbind(matrix(rnorm(150,          sd = 0.1), ncol = 3),
           matrix(rnorm(150, mean = 1, sd = 0.1), ncol = 3),
           matrix(rnorm(150, mean = 2, sd = 0.1), ncol = 3),
           matrix(rnorm(150, mean = 3, sd = 0.1), ncol = 3))

## Slightly faster way to use pam (see below)
pam1 <- function(x,k) list(cluster = pam(x,k, cluster.only=TRUE))

## We do not recommend using hier.clustering here, but if you want,
## there is factoextra::hcut () or a cheap version of it
hclusCut <- function(x, k, d.meth = "euclidean", ...)
  list(cluster = cutree(hclust(dist(x, method=d.meth), ...), k=k))

## You can manually set it before running this :   doExtras <- TRUE # or FALSE
if(!(exists("doExtras") && is.logical(doExtras)))
  doExtras <- cluster::doExtras()

if(doExtras) {
  ## Note we use B = 60 in the following examples to keep them "speedy".
  ## ---- rather keep the default B = 500 for your analysis!

  ## note we can pass 'nstart = 20' to kmeans() :
  gskmn <- clusGap(x, FUNcluster = kmeans, nstart = 20, K.max = 8, B = 60)
  gskmn #-> its print() method
  plot(gskmn, main = "clusGap(., FUNcluster = kmeans, n.start=20, B= 60)")
  set.seed(12); system.time(
    gsPam0 <- clusGap(x, FUNcluster = pam, K.max = 8, B = 60)
  )
  set.seed(12); system.time(
    gsPam1 <- clusGap(x, FUNcluster = pam1, K.max = 8, B = 60)
  )
}
```



```

## and show that it gives the "same":
not.eq <- c("call", "FUNcluster"); n <- names(gsPam0)
eq <- n[!(n %in% not.eq)]
stopifnot(identical(gsPam1[eq], gsPam0[eq]))
print(gsPam1, method="globalSEmax")
print(gsPam1, method="globalmax")

print(gsHc <- clusGap(x, FUNcluster = hclusCut, K.max = 8, B = 60))

}# end {doExtras}

gs.pam.RU <- clusGap(ruspini, FUNcluster = pam1, K.max = 8, B = 60)
gs.pam.RU
plot(gs.pam.RU, main = "Gap statistic for the 'ruspini' data")
mtext("k = 4 is best .. and k = 5 pretty close")

## This takes a minute..
## No clustering ==> k = 1 ("one cluster") should be optimal:
Z <- matrix(rnorm(256*3), 256,3)
gsP.Z <- clusGap(Z, FUNcluster = pam1, K.max = 8, B = 200)
plot(gsP.Z, main = "clusGap(<iid_rnorm_p=3>) ==> k = 1 cluster is optimal")
gsP.Z

```

clusplot

partition

clusplot(x, ...)

S3 method for class 'partition'
clusplot(x, main = NULL, dist = NULL, ...)

x	"partition" pam clarafanny
main	NULLx\$call
dist	xdisssdata pam (dist(*), keep.diss=FALSE)dist
...	clusplot.default dispar

clusplot.partition()[clusplot.default](#)
pamfannyclarapamfannyclusplotclaraNAclusplot()[clusplot.default](#)

partitiondefaultDistancesShading[clusplot.default](#)

[clusplot.default](#)[partition.object](#)[pampam.object](#)[claraclara.object](#)[fannyfanny.object](#)
[par](#)

```
## For more, see ?clusplot.default

## generate 25 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
clusplot(pam(x, 2))
## add noise, and try again :
x4 <- cbind(x, rnorm(25), rnorm(25))
clusplot(pam(x4, 2))
```

clusplot.default

```
## Default S3 method:
clusplot(x, clus, diss = FALSE,
         s.x.2d = mkCheckX(x, diss), stand = FALSE,
         lines = 2, shade = FALSE, color = FALSE,
         labels= 0, plotchar = TRUE,
         col.p = "dark green", col.txt = col.p,
         col.clus = if(color) c(2, 4, 6, 3) else 5, cex = 1, cex.txt = cex,
         span = TRUE,
         add = FALSE,
         xlim = NULL, ylim = NULL,
         main = paste("CLUSPLOT(", deparse1(substitute(x)),")"),
         sub = paste("These two components explain",
                     round(100 * var.dec, digits = 2), "% of the point variability."),
         xlab = "Component 1", ylab = "Component 2",
         verbose = getOption("verbose"),
         ...)
```

x	diss
	NA
	xdaisydist $n * (n - 1) / 2n$
clus	xcluspamfannyclara
diss	xx
s.x.2d	list $xn \times 2$ labsvar.dec
stand	
lines	0, 1, 2m1m2
	m1m2lines

	<i>m1m2</i>
	<i>m1m2</i>
shade	
color	
labels	
	<i>identify</i>
	<code>clusxxdiss = TRUE</code> <code>xattr(x,"Labels")</code> <i>daisy</i>
	<i>names</i> <code>clus</code>
plotchar	
span	
	<code>span=FALSE</code> <code>span=TRUE</code>
add	<code>label</code> <i>title</i> <code>sub</code>
col.p	
col.txt	<code>labels >= 2</code>
col.clus	
cex	<code>cex.txt</code>
xlimylim	<i>plot.default</i>
main	
sub	
xlabylab	
verbose	
...	<i>par</i>
	<code>clusplot</code> <i>princomp</i> (*, <code>cor = (ncol(x) > 2)</code>) <i>cmdscale</i> (*, <code>add=TRUE</code>) <code>diss</code>
Distances	<code>lines[i,j]</code> <code>lines = 0</code> <code>NA</code>
Shading	<code>NA</code>

```

color=TRUEpamcol.clus
col.pcol.txtcol.pcolor = TRUE

```

princompmdscalepamclaradaisyparidentifcov.mveclusplot.partition

```

## plotting votes.diss(dissimilarity) in a bivariate plot and
## partitioning into 2 clusters
data(votes.repub)
votes.diss <- daisy(votes.repub)
pamv <- pam(votes.diss, 2, diss = TRUE)
clusplot(pamv, shade = TRUE)
## is the same as
votes.clus <- pamv$clustering
clusplot(votes.diss, votes.clus, diss = TRUE, shade = TRUE)
## Now look at components 3 and 2 instead of 1 and 2:
str(cMDS <- cmdscale(votes.diss, k=3, add=TRUE))
clusplot(pamv, s.x.2d = list(x=cMDS$points[, c(3,2)],
                           labs=rownames(votes.repub), var.dec=NA),
          shade = TRUE, col.p = votes.clus,
          sub="", xlab = "Component 3", ylab = "Component 2")

clusplot(pamv, col.p = votes.clus, labels = 4)# color points and label ellipses
# "simple" cheap ellipses: larger than minimum volume:
# here they are *added* to the previous plot:
clusplot(pamv, span = FALSE, add = TRUE, col.clus = "midnightblue")

## Setting a small *label* size:
clusplot(votes.diss, votes.clus, diss = TRUE, labels = 3, cex.txt = 0.6)

if(dev.interactive()) { # uses identify() *interactively* :
  clusplot(votes.diss, votes.clus, diss = TRUE, shade = TRUE, labels = 1)
  clusplot(votes.diss, votes.clus, diss = TRUE, labels = 5)# ident. only points
}

## plotting iris (data frame) in a 2-dimensional plot and partitioning
## into 3 clusters.
data(iris)
iris.x <- iris[, 1:4]
cl3 <- pam(iris.x, 3)$clustering
op <- par(mfrow= c(2,2))
clusplot(iris.x, cl3, color = TRUE)
U <- par("usr")
## zoom in :
rect(0,-1, 2,1, border = "orange", lwd=2)
clusplot(iris.x, cl3, color = TRUE, xlim = c(0,2), ylim = c(-1,1))

```

```

box(col="orange",lwd=2); mtext("sub region", font = 4, cex = 2)
## or zoom out :
clusplot(iris.x, cl3, color = TRUE, xlim = c(-4,4), ylim = c(-4,4))
mtext("'super' region", font = 4, cex = 2)
rect(U[1],U[3], U[2],U[4], lwd=2, lty = 3)

# reset graphics
par(op)

```

coef.hclust

[diana](#)

$m(i)1 - m(i)$

coefHier()object\$heights

```

coefHier(object)
coef.hclust(object, ...)
## S3 method for class 'hclust'
coef(object, ...)
## S3 method for class 'twins'
coef(object, ...)

```

```

object          "hclust""twins"hclust\(.\)agnes\(.\)diana\(.\)
coef.hclustobject$heightsobject$mergeobjectmergeheights
coefHierobject$heights

...

```

[diana](#)[agnes.object](#) \$ ac[diana.object](#) \$ dc

```

data(agriculture)
aa <- agnes(agriculture)
coef(aa) # really just extracts aa$ac
coef(as.hclust(aa))# recomputes
coefHier(aa)      # ditto

```

daisy

```
metric = "gower"
```

```
daisy(x, metric = c("euclidean", "manhattan", "gower"),
      stand = FALSE, type = list(), weights = rep.int(1, p),
      warnBin = warnType, warnAsym = warnType, warnConst = warnType,
      warnType = TRUE)
```

```
x          n × p x numeric x factor ordered type NA
metric      "euclidean" "manhattan" "gower"

          "gower" x
stand     x
          x stand range metric
type      x
          "asymm" "A" Types dissimilarity.object
          "symm"  "S"
          "factor" factor "N" type = "S"
          "ordered" ordered "0" dissimilarity.object
          "logratio" log10 "I"
          "ordratio" ordered unclass(as.ordered(x[,j])) "T"
          "numeric" "integer" integer x "I" Types dissimilarity.object
          x
          type type = list()
weights     p ncol(x) metric = "gower" x[,k] 1
warnBin warnAsym warnConst
```

warnType

daisy dist daisy

```
x metric stand metric = "gower"[0,1]
```

symm

```
daisy type "asymm" warnBin = FALSE
```

daisy

$$\text{metric} \text{ "gower" } n_g \sqrt{p/n_g} p = n_g p / n_g n_g = 0$$

```
metric = "gower"
```

$$d_{ij} = d(i, j) = \frac{\sum_{k=1}^p w_k \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^p w_k \delta_{ij}^{(k)}}.$$

```
d_ij d_ij^(k) w_k delta_ij^(k) w_k = weights[k] delta_ij^(k) d_ij^(k) x[i,k] x[j,k]
delta_ij^(k) x[,k]
d_ij^(k) 1:K
d_ij^(k) [0,1] d_ij w_k delta_ij^(k) NA
```

```
"dissimilarity" x pam fanny agnes diana dissimilarity.object
```

```
NA type metric = "gower" weights
```

```
dissimilarity.object dist pam fanny clara agnes diana
```

```
data(agriculture)
## Example 1 in ref:
## Dissimilarities using Euclidean metric and without standardization
d.agr <- daisy(agriculture, metric = "euclidean", stand = FALSE)
d.agr
as.matrix(d.agr)[,"DK"] # via as.matrix.dist(.)
## compare with
as.matrix(daisy(agriculture, metric = "gower"))

## Example 2 in reference, extended --- different ways of "mixed" / "gower":

example(flower) # -> data(flower) *and* provide 'flowerN'

summary(d0 <- daisy(flower)) # -> the first 3 {0,1} treated as *N*ominal
summary(dS123 <- daisy(flower, type = list(symm = 1:3))) # first 3 treated as *S*ymmetric
stopifnot(dS123 == d0) # i.e., *S*ymmetric <==> *N*ominal {for 2-level factor}
summary(dNS123 <- daisy(flowerN, type = list(symm = 1:3)))
stopifnot(dS123 == d0)
## by default, however ...
summary(dA123 <- daisy(flowerN)) # .. all 3 logicals treated *A*symmetric binary (w/ warning)
summary(dA3 <- daisy(flower, type = list(asymm = 3)))
summary(dA13 <- daisy(flower, type = list(asymm = c(1, 3), ordratio = 7)))
## Mixing variable *names* and column numbers (failed in the past):
```

```
summary(df13 <- daisy(flower, type = list(asymm = c("V1", "V3"), symm= 2,
                                         ordratio= 7, logratio= 8)))

## If we'd treat the first 3 as simple {0,1}
Nflow <- flower
Nflow[,1:3] <- lapply(flower[,1:3], function(f) as.integer(as.character(f)))
summary(dN <- daisy(Nflow)) # w/ warning: treated binary .. 1:3 as interval
## Still, using Euclidean/Manhattan distance for {0-1} *is* identical to treating them as "N" :
stopifnot(dN == d0)
stopifnot(dN == daisy(Nflow, type = list(symm = 1:3))) # or as "S"
```

diana

diana

```
diana(x, diss = inherits(x, "dist"), metric = "euclidean", stand = FALSE,
      stop.at.k = FALSE,
      keep.diss = n < 100, keep.data = !diss, trace.lev = 0)
```

x	diss
	NA
	xdaisydist
diss	distdissimilarityxx
metric	
	x
stand	xx
stop.at.k	$FALSEk\{1, 2, \dots, n\}$ diana
keep.diss	keep.data
	xFALSE
trace.lev	0

```
dianadianadiana.objectplot.diana
diana
```

```
"diana"printsummaryplot
"diana""twins"pltreedianaas.hclustas.dendrogram
diana
order
order.lab      order
```



```

height
dc           $d(i)dc1 - d(i)dc$ 
merge        $mergej|j|$ 
diss        "dissimilarity"
data        standagnes

```

[agnes.cutree](#) as [hclust](#) [daisy](#) [dist](#) [plot](#) [diana](#) [twins](#) .object

```

data(votes.repub)
dv <- diana(votes.repub, metric = "manhattan", stand = TRUE)
print(dv)
plot(dv) #-> plot.diana() {w/ its own help + examples}

## Cut into 2 groups:
dv2 <- cutree(as.hclust(dv), k = 2)
table(dv2) # 8 and 42 group members
rownames(votes.repub)[dv2 == 1]

## For two groups, does the metric matter ?
dv0 <- diana(votes.repub, stand = TRUE) # default: Euclidean
dv.2 <- cutree(as.hclust(dv0), k = 2)
table(dv2 == dv.2) ## identical group assignments

str(as.dendrogram(dv0)) # {via as.dendrogram.twins() method}

data(agriculture)
## Plot similar to Figure 8 in ref
## Not run: plot(diana(agriculture), ask = TRUE)

```

dissimilarity.object

"dissimilarity"

```

donn <- attr(do, "Size")  $i < j \leq ndo[n*(i-1) - i*(i-1)/2 + j-i]n * (n-1)/2n^2$ 
"dissimilarity" dist dist as.matrix  $d_{ij}$  as.matrix(do)[i,j]

```

```

Size
Metric
Labels
NA.message
Types      typedaisy()
           A

```

```

S
N
O
I "logratio"
T ordered

```

[daisy](#)[pam](#)[clarafannyagnesdianadissimilarity](#)

"dissimilarity"[printsummary](#)

[daisydistpamclarafannyagnesdiana](#)

[ellipsoidhull](#)

```

ellipsoidhull(x, tol=0.01, maxit=5000,
              ret.wt = FALSE, ret.sqdist = FALSE, ret.pr = FALSE)
## S3 method for class 'ellipsoid'
print(x, digits = max(1, getOption("digits") - 2), ...)

```

```

x                 $npn \times p$ 
tol              maxit
maxit
ret.wtret.sqdistret.pr
                ret.wtret.sqdistret.pr
digits...       print

```

[clusplot.default](#)
[cov.mve](#)MASS

"ellipsoid"[list](#)

```

cov               $p \times p$ 
loc               $p$ 
d2                $d2 = t^2 t$ ellipsed2 = qchisq(alpha, df = p)alpha $\text{loc}$ cov
wt              ret.wt
sqdist          ret.sqdist
prob            ret.pr
it
tolmaxit
eps              $p$ 
ierr            0
conv            it < maxit && ierr == 0

```

[predict.ellipsoid](#)
[predictellipsoid](#)
[volume.ellipsoid](#)
[ellipsoid](#)
[ellipsee](#)
[ellipse](#)
[Points](#)
[chullclus](#)
[plotcov.mve](#)

```

x <- rnorm(100)
xy <- unname(cbind(x, rnorm(100) + 2*x + 10))
exy. <- ellipsoidhull(xy)
exy. # >> calling print.ellipsoid()

plot(xy, main = "ellipsoidhull(<Gauss data>) -- 'spanning points'")
lines(predict(exy.), col="blue")
points(rbind(exy.$loc), col = "red", cex = 3, pch = 13)

exy <- ellipsoidhull(xy, tol = 1e-7, ret.wt = TRUE, ret.sqdist = TRUE)
str(exy) # had small 'tol', hence many iterations
(ii <- which(zapsmall(exy $ wt) > 1e-6))
## --> only about 4 to 6 "spanning ellipsoid" points
round(exy$wt[ii],3); sum(exy$wt[ii]) # weights summing to 1
points(xy[ii,], pch = 21, cex = 2,
       col="blue", bg = adjustcolor("blue",0.25))

```

fanny

k

```

fanny(x, k, diss = inherits(x, "dist"), memb.exp = 2,
      metric = c("euclidean", "manhattan", "SqEuclidean"),
      stand = FALSE, iniMem.p = NULL, cluster.only = FALSE,
      keep.diss = !diss && !cluster.only && n < 100,
      keep.data = !diss && !cluster.only,
      maxit = 500, tol = 1e-15, trace.lev = 0)

```

x diss

[xdaisydist](#)

k $0 < k < n/2n$

```

diss            distdissimilarityxx
memb.exp        r2
metric          "euclidean""manhattan""SqEuclidean""SqEuclidean"
                x
stand           xx
iniMem.p         $n \times k$ NULLmembership
cluster.only
keep.disskeep.data
                xFALSE
maxittol        maxit = 500tol = 1e-15
trace.lev       0

```

$u_{iv}i v$

fannydaisy**memb.exp**iniMem.p**maxittol**

$$\sum_{v=1}^k \frac{\sum_{i=1}^n \sum_{j=1}^n u_{iv}^r u_{jv}^r d(i, j)}{2 \sum_{j=1}^n u_{jv}^r}$$

nkr **memb.exp** $d(i, j)ij$

$r \rightarrow 1r \rightarrow \infty r = 2u_{iv} \equiv 1/k$ **memb.exp** $= r$

fannyspherical cluster**plot.partition**

"fanny"**fanny.object**

agnesfanny.objectpartition.objectplot.partitiondaisydist

```

## generate 10+15 objects in two clusters, plus 3 objects lying
## between those clusters.
x <- rbind(cbind(rnorm(10, 0, 0.5), rnorm(10, 0, 0.5)),
           cbind(rnorm(15, 5, 0.5), rnorm(15, 5, 0.5)),
           cbind(rnorm( 3,3.2,0.5), rnorm( 3,3.2,0.5)))
fannyx <- fanny(x, 2)
## Note that observations 26:28 are "fuzzy" (closer to # 2):
fannyx
summary(fannyx)
plot(fannyx)

(fan.x.15 <- fanny(x, 2, memb.exp = 1.5)) # 'crispier' for obs. 26:28
(fanny(x, 2, memb.exp = 3))              # more fuzzy in general

data(ruspini)
f4 <- fanny(ruspini, 4)
stopifnot(rle(f4$clustering)$lengths == c(20,23,17,15))
plot(f4, which = 1)
## Plot similar to Figure 6 in Stryuf et al (1996)
plot(fanny(ruspini, 5))

```

fanny.object

"fanny"

fanny

membership

memb.exp

coeff
$$\frac{F(k)kF(k)1/k}{(F(k) - 1/k)/(1 - 1/k)}$$

clustering [partition.object](#)

k.crisp $\leq k$ memb.exp

objective tol

convergence iterationsconvergedmaxittol

diss "dissimilarity"[partition.object](#)

call [partition.object](#)

silinfo [partition.object](#)

data [partition.object](#)

[fanny](#)

"fanny"printsummary

"fanny""partition"plotclusplotfanny

[fannyprint.fannydissimilarity.objectpartition.objectplot.partition](#)

flower

data(flower)

agnes

```
data(flower)
str(flower) # factors, ordered, numeric

## "Nicer" version (less numeric more self explainable) of 'flower':
flowerN <- flower
colnames(flowerN) <- c("winters", "shadow", "tubers", "color",
                      "soil", "preference", "height", "distance")
for(j in 1:3) flowerN[,j] <- (flowerN[,j] == "1")
levels(flowerN$color) <- c("1" = "white", "2" = "yellow", "3" = "pink",
                          "4" = "red", "5" = "blue")[levels(flowerN$color)]
levels(flowerN$soil) <- c("1" = "dry", "2" = "normal", "3" = "wet")[levels(flowerN$soil)]
flowerN

## ==> example(daisy) on how it is used
```

lower.to.upper.tri.inds

```
lower.to.upper.tri.inds(n)
upper.to.lower.tri.inds(n)
```

n

$$1:NN = n(n-1)/2$$

[upper.tri](#)[lower.tri](#)

```
m5 <- matrix(NA,5,5)
m <- m5; m[lower.tri(m)] <- upper.to.lower.tri.inds(5); m
m <- m5; m[upper.tri(m)] <- lower.to.upper.tri.inds(5); m

stopifnot(lower.to.upper.tri.inds(2) == 1,
           lower.to.upper.tri.inds(3) == 1:3,
           upper.to.lower.tri.inds(3) == 1:3,
           sort(upper.to.lower.tri.inds(5)) == 1:10,
           sort(lower.to.upper.tri.inds(6)) == 1:15)
```

medoids	pam
-------------------------	---------------------

[x](#)[pam\(\)](#)

[medoids\(x, clustering, diss = inherits\(x, "dist"\), USE.NAMES = FALSE, ...\)](#)

x	pam
clustering	nclustering1:k partition.object cutree()
diss	pam
USE.NAMES	vapply()
...	pam(xj, k=1, ...) metricvariant="f_5" trace.lev = k

[pam\(\)](#)[kmeans\(\)](#)

[pamkmeanscutree\(\)](#)[agneshclust](#)

```
## From example(agnes):
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)
agn2 <- agnes(daisy(votes.repub), diss = TRUE, method = "complete")
agnS <- agnes(votes.repub, method = "flexible", par.method = 0.625)

for(k in 2:11) {
  print(table(cl.k <- cutree(agnS, k=k)))
  stopifnot(length(cl.k) == nrow(votes.repub), 1 <= cl.k, cl.k <= k, table(cl.k) >= 2)
  m.k <- medoids(votes.repub, cl.k)
  cat("k =", k, "; sort(medoids) = "); dput(sort(m.k), control={})
}
```

mona

```
mona(x, trace.lev = 0)
```

```
x          NANA
trace.lev
```

```
monaagnesdiana
mona
```

```
"mona"mona.object
```

NA

```
mona(x)xNA
```

```
ncol(x) == 1
```

```
agnesmona.objectplot.mona
```



```

data(animals)
ma <- mona(animals)
ma
## Plot similar to Figure 10 in Struyf et al (1996)
plot(ma)

## One place to see if/how error messages are *translated* (to 'de' / 'pl'):
ani.NA <- animals; ani.NA[4,] <- NA
aniNA <- within(animals, { end[2:9] <- NA })
aniN2 <- animals; aniN2[cbind(1:6, c(3, 1, 4:6, 2))] <- NA
ani.non2 <- within(animals, end[7] <- 3 )
ani.idNA <- within(animals, end[!is.na(end)] <- 1 )
try( mona(ani.NA) ) ## error: .. object with all values missing
try( mona(aniNA) ) ## error: .. more than half missing values
try( mona(aniN2) ) ## error: all have at least one missing
try( mona(ani.non2) ) ## error: all must be binary
try( mona(ani.idNA) ) ## error: ditto

```

mona.object

"mona"[mona](#)

mona

data

order

order.lab order

variable order

step order

"mona"printsummaryplot

[monaplot.mona](#)

pam

k

```
pam(x, k, diss = inherits(x, "dist"),
    metric = c("euclidean", "manhattan"),
    medoids = if(is.numeric(nstart)) "random",
    nstart = if(variant == "faster") 1 else NA,
    stand = FALSE, cluster.only = FALSE,
    do.swap = TRUE,
    keep.diss = !diss && !cluster.only && n < 100,
    keep.data = !diss && !cluster.only,
    variant = c("original", "o_1", "o_2", "f_3", "f_4", "f_5", "faster"),
    pamonce = FALSE, trace.lev = 0)
```

x	diss
	NA
	xdaisydistNA
k	
diss	distdissimilarityxx
metric	
	x
medoids	k1:n
nstart	medoids = "random"kmeans()
stand	xx
cluster.only	
do.swap	TRUEndo.swap = FALSE
keep.diss	keep.data
	xFALSE
pamonce	0:6
variant	characterpamonce"faster"pamonce = 6nstart = 1medoids = "random"
trace.lev	0

```
pamkmeanspamplot.partitionmean(silhouette(pr)[, "sil_width"])pr <- pam(..)
pr$silinfo$avg.widthpam.object
cluster.only
pam(x,k, cluster.only=TRUE)
pam(x,k)$clustering
pamkkkk
medoids
medoids
```

```
pamonce
FALSE0pamonce = 1TRUEpamonce = 2
```

```
pamonce = 3
pamonce = 4
pamonce = 5 pamonce = 2
```

```
pamonce = 6  $O(n^2)O(n(n-k)k)k$ 
 $O(n^2k)$ 
nstart > 1
```

```
"pam"?pam.object
```

```
pam $O(n^2)$ clara()
 $n \leq 655362^{16}nn(n-1)/2$ .Machine$integer.max $2^{31} - 1$ 
```

```
cluster.only=TRUEdo.swap=FALSE
pamonce12
pamonce36
```

<https://arxiv.org/abs/2008.05171>

[agnes](#)[pam](#)[object](#)[claradaisy](#)[partition](#)[object](#)[plot](#)[partition](#)[dist](#)

```
## generate 25 objects, divided into 2 clusters.
set.seed(17) # to get reproducible data:
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
pamx <- pam(x, 2)
pamx # Medoids: '9' and '15' ...
summary(pamx)
plot(pamx)
stopifnot(pamx$id.med == c(9, 15))
stopifnot(identical(pamx$clustering, rep(1:2, c(10, 15))))

## use obs. 1 & 16 as starting medoids -- same result (for seed above, *and* typically) :
(p2m <- pam(x, 2, medoids = c(1,16)))
## no _build_ *and* no _swap_ phase: just cluster all obs. around (1, 16):
p2.s <- pam(x, 2, medoids = c(1,16), do.swap = FALSE)
```

```

p2.s
keep_nms <- setdiff(names(pamx), c("call", "objective"))# .$objective["build"] differ
stopifnot(p2.s$id.med == c(1,16), # of course
          identical(pamx[keep_nms],
                    p2m[keep_nms]))

p3m <- pam(x, 3, trace.lev = 2)
## rather stupid initial medoids:
(p3m. <- pam(x, 3, medoids = 3:1, trace.lev = 1))

pam(daisy(x, metric = "manhattan"), 2, diss = TRUE)

data(ruspini)
## Plot similar to Figure 4 in Stryuf et al (1996)
## Not run: plot(pam(ruspini, 4), ask = TRUE)

```

pam.object

"pam"

pamlist

medoids	pammedoidsmatrix
id.med	
clustering	partition.object
objective	pam
isolation	

clusinfo

silinfo	partition.object
diss	partition.object
call	partition.object
data	partition.object

pam

"pam"printsummary

"pam""partition"plotclusplotpam

pamdissimilarity.objectpartition.objectplot.partition

```
## Use the silhouette widths for assessing the best number of clusters,
## following a one-dimensional example from Christian Hennig :
##
x <- c(rnorm(50), rnorm(50,mean=5), rnorm(30,mean=15))
asw <- numeric(20)
## Note that "k=1" won't work!
for (k in 2:20)
  asw[k] <- pam(x, k) $ silinfo $ avg.width
k.best <- which.max(asw)
cat("silhouette-optimal number of clusters:", k.best, "\n")

plot(1:20, asw, type= "h", main = "pam() clustering assessment",
     xlab= "k (# clusters)", ylab = "average silhouette width")
axis(1, k.best, paste("best",k.best,sep="\n"), col = "red", col.axis = "red")
```

partition.object

"partition"

"partition"

clustering	n
call	<code>call</code>
silinfo	$1 < k < n$ <code>silhouette</code> <code>silhouette()</code> $s(i)$
	$s(i)$ <code>plot.partition</code> avg.width
objective	
diss	"dissimilarity" clara
data	

pamclarafanny

"partition"plotclusplot

"partition""pam""clara""fanny"
pam.objectclara.objectfanny.object

pamclarafanny

plantTraits

data(plantTraits)

pdias
longindex
durflow
height 128
begflow 123456789
mycor 012
vegaer 012
vegsout vegaer
autopoll 0123
insects 04
wind 04
lign 0:1
piq
ros
semiros 01
leafy
suman
winan
monocarp
polycarp
seasaes
seashiv
seasver
everalw
everparti
elaio
endozoo
epizoo
aquat
windgl
unsp

```

data(plantTraits)

## Calculation of a dissimilarity matrix
library(cluster)
dai.b <- daisy(plantTraits,
              type = list(ordratio = 4:11, symm = 12:13, asymm = 14:31))

## Hierarchical classification
agn.trts <- agnes(dai.b, method="ward")
plot(agn.trts, which.plots = 2, cex= 0.6)
plot(agn.trts, which.plots = 1)
cutree6 <- cutree(agn.trts, k=6)
cutree6

## Principal Coordinate Analysis
cmds dai.b <- cmdscale(dai.b, k=6)
plot(cmdsdai.b[, 1:2], asp = 1, col = cutree6)

```

plot.agnes

agnes

```

## S3 method for class 'agnes'
plot(x, ask = FALSE, which.plots = NULL, main = NULL,
      sub = paste("Agglomerative Coefficient = ",round(x$ac, digits = 2)),
      adj = 0, nmax.lab = 35, max.strlen = 5, xax.pretty = TRUE, ...)

```

x	"agnes" agnes(.)
ask	which.plotsNULL plot.agnesmenu
which.plots	which.plots12
main	plot.default
adj	bannerplot()
nmax.lab	
max.strlen	
xax.pretty	pretty(*, n = xax.pretty) xax.pretty = FALSE
...	parbannerplot() pltree() pltree.twins

```
ask = TRUEplot.agnespar(ask= TRUE)
agnesheight
```

```
bannerplotpltree()pltree.twins
xlabylab
```

```
nmax.labmax.strlen
pltree()dg <- as.dendrogram(x)dgplot.dendrogram
```

```
agnesagnes.objectbannerplotpltree.twinspar
```

```
## Can also pass 'labels' to pltree() and bannerplot():
data(iris)
cS <- as.character(Sp <- iris$Species)
cS[Sp == "setosa"] <- "S"
cS[Sp == "versicolor"] <- "V"
cS[Sp == "virginica"] <- "G"
ai <- agnes(iris[, 1:4])
plot(ai, labels = cS, nmax.lab = 150)# bannerplot labels are mess
```

```
plot.diana
```

```
diana
```

```
## S3 method for class 'diana'
plot(x, ask = FALSE, which.plots = NULL, main = NULL,
      sub = paste("Divisive Coefficient = ", round(x$dc, digits = 2)),
      adj = 0, nmax.lab = 35, max.strlen = 5, xax.pretty = TRUE, ...)
```



```

x            "diana"diana\(.\)
ask          which.plotsNULLplot.dianamenu
which.plots  which.plots12
mainsub     plot.default
adj         bannerplot\(\)
nmax.lab
max.strlen
xax.pretty  pretty(*, n = xax.pretty)xax.pretty = FALSE
...        parbannerplot\(\)pltree\(\)

```

```

ask = TRUEplot.dianapar(ask= TRUE)
dianaheight

```

```

nmax.labmax.strlen

```

```

plot.agnes

```

```

dianadiana.objecttwins.objectpar

```

```

example(diana)# -> dv <- diana(...)

plot(dv, which.plots = 1, nmax.lab = 100)

## wider labels :
op <- par(mar = par("mar") + c(0, 2, 0,0))
plot(dv, which.plots = 1, nmax.lab = 100, max.strlen = 12)
par(op)

```

plot.mona

mona

```
## S3 method for class 'mona'
plot(x, main = paste("Banner of ", deparse1(x$call)),
      sub = NULL, xlab = "Separation step",
      col = c(2,0), axes = TRUE, adj = 0,
      nmax.lab = 35, max.strlen = 5, ...)
```

x	"mona" mona(.)
mainsub	plot.default
xlab	title
coladj	bannerplot()
axes	
nmax.lab	
max.strlen	
...	bannerplot() text

monastep

nmax.labmax.strlen

[plot.agnes](#)

[monamona.objectpar](#)

plot.partition

partition

```
## S3 method for class 'partition'
plot(x, ask = FALSE, which.plots = NULL,
      nmax.lab = 40, max.strlen = 5, data = x$data, dist = NULL,
      stand = FALSE, lines = 2,
      shade = FALSE, color = FALSE, labels = 0, plotchar = TRUE,
      span = TRUE, xlim = NULL, ylim = NULL, main = NULL, ...)
```

```
x          "partition" pam clara fanny
ask         which.plots NULL plot.partition menu
which.plots which.plots 12
nmax.lab
max.strlen
data        x
dist        xdis pam(*, keep.diss=FALSE) dist
stand lines shade color labels plotchar span xlim ylim main ...
          clusplot.default diss par
```

```
ask= TRUE plot.partition par(ask= TRUE)
clusplot.partition
silhouette
kk
```

```
nmax.lab max.strlen
plot(silhouette(x), ...) plot.silhouette
```

[plot.agnes](#)

[partition.object](#) [clusplot.partition](#) [clusplot.default](#) [pam](#) [pam.object](#) [clara](#)
[clara.object](#) [fanny](#) [fanny.object](#) [par](#)

```
## generate 25 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
plot(pam(x, 2))

## Save space not keeping data in clus.object, and still clusplot() it:
data(xclara)
cx <- clara(xclara, 3, keep.data = FALSE)
cx$data # is NULL
plot(cx, data = xclara)
```

pltree

```
pltree()twinstwinsagnes()diana\(\)
```

```
pltree(x, ...)
## S3 method for class 'twins'
pltree(x, main = paste("Dendrogram of ", deparse1(x$call)),
       labels = NULL, ylab = "Height", ...)
```

```
x          pltree"twins"agnes\(\)diana\(\)
main
labels      x
ylab
...         par
```

```
twins
plot(as.hclust(x), ...)plot.hclust(..)xx <- as.dendrogram(as.hclust(x))xx
plot.dendrogram
```

[agnesagnes.objectdianadiana.objecthclustparplot.agnesplot.diana](#)

```
data(votes.repub)
agn <- agnes(votes.repub)
pltree(agn)

dagn <- as.dendrogram(as.hclust(agn))
dagn2 <- as.dendrogram(as.hclust(agn), hang = 0.2)
op <- par(mar = par("mar") + c(0,0,0, 2)) # more space to the right
plot(dagn2, horiz = TRUE)
plot(dagn, horiz = TRUE, center = TRUE,
     nodePar = list(lab.cex = 0.6, lab.col = "forest green", pch = NA),
     main = deparse(agn$call))
par(op)
```

pluton

pluton

data(pluton)

^{238}Pu

$^{239}\text{Pu}^{238}\text{U}$

pluton.dat.../datasets/clusplot-examples.tar.gz

data(pluton)

```
hist(apply(pluton,1,sum), col = "gray") # between 94% and 100%
pu5 <- pluton
pu5$Pu242 <- 100 - apply(pluton,1,sum) # the remaining isotope.
pairs(pu5)
```

predict.ellipsoid

```
predict.ellipsoid(object, n.out=201, ...)
## S3 method for class 'ellipsoid'
predict(object, n.out=201, ...)
ellipsoidPoints(A, d2, loc, n.half = 201)
```

```

object          ellipsoidellipsoidhull()
n.outn.half
Ad2loc          ellipsoidPoints
...

```

```

ellipsoidPointspredict.ellipsoidellipsoidellipsoidhullobjectlocpp  ×  pcovAd2
qchisq(*, p)
 $p = 2p \geq 3$ 

```

```

2*n.outp

```

```

ellipsoidhullvolume.ellipsoid

```

```

## see also  example(ellipsoidhull)

## Robust vs. L.S. covariance matrix
set.seed(143)
x <- rt(200, df=3)
y <- 3*x + rt(200, df=2)
plot(x,y, main="non-normal data (N=200)")
mtext("with classical and robust cov.matrix ellipsoids")
X <- cbind(x,y)
C.ls <- cov(X) ; m.ls <- colMeans(X)
d2.99 <- qchisq(0.99, df = 2)
lines(ellipsoidPoints(C.ls, d2.99, loc=m.ls), col="green")
if(require(MASS)) {
  Cxy <- cov.rob(cbind(x,y))
  lines(ellipsoidPoints(Cxy$cov, d2 = d2.99, loc=Cxy$center), col="red")
}# MASS

```

```

print.agnes

```

```

agnes
print()agnesagnes.object

```

```

## S3 method for class 'agnes'
print(x, ...)

```

```

x
...

```

```

summary.agnesagnesagnes.objectprintprint.default

```

```
print.clara
```

```
clara
```

```
print()clara
```

```
## S3 method for class 'clara'  
print(x, ...)
```

```
x
```

```
...
```

```
summary.claraclaraclara.objectprintprint.default
```

```
print.diana
```

```
diana
```

```
print()diana
```

```
## S3 method for class 'diana'  
print(x, ...)
```

```
x
```

```
...
```

```
dianadiana.objectprintprint.default
```

```
print.dissimilarity
```

```
dissimilarity
print()summary()dissimilarityprintprint.defaultsummary

## S3 method for class 'dissimilarity'
print(x, diag = NULL, upper = NULL,
      digits = getOption("digits"), justify = "none", right = TRUE, ...)
## S3 method for class 'dissimilarity'
summary(object,
         digits = max(3, getOption("digits") - 2), ...)
## S3 method for class 'summary.dissimilarity'
print(x, ...)

xobject      dissimilaritysummary.dissimilarityprint.summary.dissimilarity()
digits       print.default
diagupperjustifyright
              print.dist
...

daisydissimilarity.objectprintprint.defaultprint.dist

## See  example(daisy)

sd <- summary(daisy(matrix(rnorm(100), 20,5)))
sd # -> print.summary.dissimilarity(.)
str(sd)
```

```
print.fanny
```

```
fanny
print()fanny

## S3 method for class 'fanny'
print(x, digits = getOption("digits"), ...)
## S3 method for class 'fanny'
summary(object, ...)
## S3 method for class 'summary.fanny'
print(x, digits = getOption("digits"), ...)
```



```
xobject      fanny
digits      print.default
...
```

```
fannyfanny.objectprintprint.default
```

```
print.mona
```

```
mona
print()mona
```

```
## S3 method for class 'mona'
print(x, ...)
```

```
x
...
```

```
monamonamona.objectprintprint.default
```

```
print.pam
```

```
pam
print()pam
```

```
## S3 method for class 'pam'
print(x, ...)
```

```
x
...
```

```
pampam.pam.objectprintprint.default
```

ruspini

```
data(ruspini)
```

agnes

```
data(ruspini)
```

```
## Plot similar to Figure 4 in Stryuf et al (1996)
## Not run: plot(pam(ruspini, 4), ask = TRUE)
```

```
## Plot similar to Figure 6 in Stryuf et al (1996)
plot(fanny(ruspini, 5))
```

silhouette

k

```
silhouette(x, ...)
## Default S3 method:
  silhouette(x, dist, dmatrix, ...)
## S3 method for class 'partition'
silhouette(x, ...)
## S3 method for class 'clara'
silhouette(x, full = FALSE, subset = NULL, ...)

sortSilhouette(object, ...)
## S3 method for class 'silhouette'
summary(object, FUN = mean, ...)
## S3 method for class 'silhouette'
plot(x, nmax.lab = 40, max.strlen = 5,
     main = NULL, sub = NULL, xlab = expression("Silhouette width " * s[i]),
     col = "gray", do.col.sort = length(col) > 1, border = 0,
     cex.names = par("cex.axis"), do.n.k = TRUE, do.clus.stat = TRUE, ...)
```

```

x          default  $k \times k$  clustering  $2 \leq k \leq n - 1$ 
dist       dist matrix
dmatrix     $n \times n$  dist
full        $[0, 1]$  clarafsample.int(n, size = f*n)  $O((f * n)^2)$  daisy
subset     1:nfull
object     silhouette
...
FUN
nmax.lab
max.strlen
mainsublab title
colbordercex.names
              barplot() col = heat.colors(n), border = par("fg")
              colkdo.col.sort
do.col.sort  col
do.n.k        $nk$ 
do.clus.stat

```

$$s(i) := 0d(i, C)d(i, C)b(i) := \min_C d(i, C)$$

$$s(i) := \frac{b(i) - a(i)}{\max(a(i), b(i))}.$$

```

silhouette.default()cluster:::silhouetteR
s(i)s(i)s(i)

```

```

silhouette()sil $\times$  3sil[i,]s(i)colnamesc("cluster", "neighbor",
"sil_width")
summary(sil)summary.silhouette
si.summary summarys(i)
clus.avg.widths mean = FUN
avg.width FUN(s)s
clus.sizes tablek
call callsil
Ordered attr(sil, "Ordered")

sortSilhouette(sil)sil $s(i)$ 
attr(sil, "Ordered")silsortSilhouette()rownames(sil)
attr(sil, "iOrd")

```

```
silhouette()partition silhouette.default()cutree()
clara()full = TRUE
```

```
plot.agnes
```

```
partition.objectplot.partition
```

```
data(ruspini)
pr4 <- pam(ruspini, 4)
str(si <- silhouette(pr4))
(ssi <- summary(si))
plot(si) # silhouette plot
plot(si, col = c("red", "green", "blue", "purple"))# with cluster-wise coloring

si2 <- silhouette(pr4$clustering, dist(ruspini, "canberra"))
summary(si2) # has small values: "canberra"'s fault
plot(si2, nmax= 80, cex.names=0.6)

op <- par(mfrow= c(3,2), oma= c(0,0, 3, 0),
          mgp= c(1.6,.8,0), mar= .1+c(4,2,2,2))
for(k in 2:6)
  plot(silhouette(pam(ruspini, k=k)), main = paste("k = ",k), do.n.k=FALSE)
mtext("PAM(Ruspini) as in Kaufman & Rousseeuw, p.101",
      outer = TRUE, font = par("font.main"), cex = par("cex.main")); frame()

## the same with cluster-wise colours:
c6 <- c("tomato", "forest green", "dark blue", "purple2", "goldenrod4", "gray20")
for(k in 2:6)
  plot(silhouette(pam(ruspini, k=k)), main = paste("k = ",k), do.n.k=FALSE,
       col = c6[1:k])
par(op)

## clara(): standard silhouette is just for the best random subset
data(xclara)
set.seed(7)
str(xc1k <- xclara[ sample(nrow(xclara), size = 1000) ,]) # rownames == indices
cl3 <- clara(xc1k, 3)
plot(silhouette(cl3))# only of the "best" subset of 46
## The full silhouette: internally needs large (36 MB) dist object:
sf <- silhouette(cl3, full = TRUE) ## this is the same as
s.full <- silhouette(cl3$clustering, daisy(xc1k))
stopifnot(all.equal(sf, s.full, check.attributes = FALSE, tolerance = 0))
## color dependent on original "3 groups of each 1000": % __FIXME ??__
plot(sf, col = 2+ as.integer(names(cl3$clustering) ) %/% 1000,
     main ="plot(silhouette(clara(.), full = TRUE))")

## Silhouette for a hierarchical clustering:
ar <- agnes(ruspini)
si3 <- silhouette(cutree(ar, k = 5), # k = 4 gave the same as pam() above
```

```

      daisy(ruspini))
stopifnot(is.data.frame(di3 <- as.data.frame(si3)))
plot(si3, nmax = 80, cex.names = 0.5)
## 2 groups: Agnes() wasn't too good:
si4 <- silhouette(cutree(ar, k = 2), daisy(ruspini))
plot(si4, nmax = 80, cex.names = 0.5)

```

sizeDiss

$$f(n) = n(n-1)/2$$

sizeDiss(d)

d $n(n-1)/2$

nlength(d) == n(n-1)/2NA

[dissimilarity.object](#)[as.dist](#)dissimilaritydistSize

```

sizeDiss(1:10)# 5, since 10 == 5 * (5 - 1) / 2
sizeDiss(1:9) # NA

```

```

n <- 1:100
stopifnot(n == sapply( n*(n-1)/2, function(n) sizeDiss(logical(n))))

```

summary.agnes

[agnes](#)[print.agnes](#)

```

## S3 method for class 'agnes'
summary(object, ...)
## S3 method for class 'summary.agnes'
print(x, ...)

```

xobject [agnes](#)

...

[agnes](#)[agnes.object](#)

```

data(agriculture)
summary(agnes(agriculture))

```

summary.clara

claraprint.clara

```
## S3 method for class 'clara'
summary(object, ...)
## S3 method for class 'summary.clara'
print(x, ...)
```

xobject clara

...

clara.object

```
## generate 2000 objects, divided into 5 clusters.
set.seed(47)
x <- rbind(cbind(rnorm(400, 0,4), rnorm(400, 0,4)),
           cbind(rnorm(400,10,8), rnorm(400,40,6)),
           cbind(rnorm(400,30,4), rnorm(400, 0,4)),
           cbind(rnorm(400,40,4), rnorm(400,20,2)),
           cbind(rnorm(400,50,4), rnorm(400,50,4))
)
clx5 <- clara(x, 5)
## Mis'classification' table:
table(rep(1:5, rep(400,5)), clx5$clustering) # -> 1 "error"
summary(clx5)

## Graphically:
par(mfrow = c(3,1), mgp = c(1.5, 0.6, 0), mar = par("mar") - c(0,0,2,0))
plot(x, col = rep(2:6, rep(400,5)))
plot(clx5)
```

summary.diana

diana

```
## S3 method for class 'diana'
summary(object, ...)
## S3 method for class 'summary.diana'
print(x, ...)
```

xobject diana

...

dianadiana.object

summary.mona

mona

```
## S3 method for class 'mona'
summary(object, ...)
## S3 method for class 'summary.mona'
print(x, ...)
```

xobject mona

...

monamona.object

summary.pam

pamsummary.pamprint

```
## S3 method for class 'pam'
summary(object, ...)
## S3 method for class 'summary.pam'
print(x, ...)
```

xobject pam

...

pampam.object

twins.object

"twins"

agnes.objectdiana.object

agnesdiana

"twins"pltree

"twins""agnes""diana"

agnesdiana

volume.ellipsoid

ellipsoidellipsoidhull()

```
volume(object, ...)
## S3 method for class 'ellipsoid'
volume(object, log = FALSE, ...)
```

object	ellipsoidellipsoidhull
log	logical
...	log

$V \log(V)$ log = TRUEobject

clusplot $d > 2$

ellipsoidhull


```
## example(ellipsoidhull) # which defines 'ellipsoid' object <namefoo>

myEl <- structure(list(cov = rbind(c(3,1),1:2), loc = c(0,0), d2 = 10),
                    class = "ellipsoid")
volume(myEl)# i.e. "area" here (d = 2)
myEl # also mentions the "volume"

set.seed(1)
d5 <- matrix(rt(500, df=3), 100,5)
e5 <- ellipsoidhull(d5)
```

votes.repub

data(votes.repub)

xclara

data(xclara)

V1V2 xy

```
xclararead.table("xclara.dat")all.equal1.15e-7V11.17e-7V2options(digits = 7)
pam(*, 3){1 : 900}{901 : 2050}{2051 : 3000}
```

xclara.datclus_examples.tar.gz

```

## Visualization: Assuming groups are defined as {1:1000}, {1001:2000}, {2001:3000}
plot(xclara, cex = 3/4, col = rep(1:3, each=1000))
p.ID <- c(78, 1411, 2535) ## PAM's medoid indices == pam(xclara, 3)$id.med
text(xclara[p.ID,], labels = 1:3, cex=2, col=1:3)

px <- pam(xclara, 3) ## takes ~2 seconds
cxcl <- px$clustering ; iCl <- split(seq_along(cxcl), cxcl)
boxplot(iCl, range = 0.7, horizontal=TRUE,
        main = "Indices of the 3 clusters of pam(xclara, 3)")

## Look more closely now:
bxCl <- boxplot(iCl, range = 0.7, plot=FALSE)
## We see 3 + 2 + 2 = 7 clear "outlier"s or "wrong group" observations:
with(bxCl, rbind(out, group))
## out   1038 1451 1610   30  327  562  770
## group    1    1    1    2    2    3    3
## Apart from these, what are the robust ranges of indices? -- Robust range:
t(iR <- bxCl$stats[c(1,5),])
##      1  900
##    901 2050
##  2051 3000
gc <- adjustcolor("gray20",1/2)
abline(v = iR, col = gc, lty=3)
axis(3, at = c(0, iR[2,]), padj = 1.2, col=gc, col.axis=gc)

```


codetools

checkUsage

```
checkUsage(fun, name = "<anonymous>", report = cat, all = FALSE,
           suppressLocal = FALSE, suppressParamAssigns = !all,
           suppressParamUnused = !all, suppressFundefMismatch = FALSE,
           suppressLocalUnused = FALSE, suppressNoLocalFun = !all,
           skipWith = FALSE, suppressUndefined = dfltSuppressUndefined,
           suppressPartialMatchArgs = TRUE)
checkUsageEnv(env, ...)
checkUsagePackage(pack, ...)
```

```
fun
name
env
pack
...          checkUsage
report
all
suppressLocal
suppressParamAssigns

suppressParamUnused

suppressFundefMismatch

suppressLocalUnused
```

suppressNoLocalFun

skipWith within

suppressUndefined

suppressPartialMatchArgs

checkUsagesuppressLocal=TRUEsuppressXYZ

checkUsageEnvcheckUsagePackagecheckUsagecheckUsagePackage

```
checkUsage(checkUsage)
checkUsagePackage("codetools",all=TRUE)
## Not run: checkUsagePackage("base",suppressLocal=TRUE)
```

codetools

```
collectLocals(e, collect)
collectUsage(fun, name = "<anonymous>", ...)
constantFold(e, env = NULL, fail = NULL)
findFuncLocals(formals, body)
findLocals(e, envir = .BaseEnv)
findLocalsList(elist, envir = .BaseEnv)
flattenAssignment(e)
getAssignedVar(e)
isConstantValue(v, w)
makeCodeWalker(..., handler, call, leaf)
makeConstantFolder(..., leaf, handler, call, exit, isLocal, foldable,
                    isConstant, signal)
makeLocalsCollector(..., leaf, handler, isLocal, exit, collect)
makeUsageCollector(fun, ..., name, enterLocal, enterGlobal, enterInternal,
                   startCollectLocals, finishCollectLocals, warn,
                   signal)
walkCode(e, w = makeCodeWalker())
```

e
elist
v
fun
formals
body
name
env
envir
w
...
collect
fail
handler
call
leaf
isLocal
exit
enterLocal
enterGlobal
enterInternal
startCollectLocals

finishCollectLocals

warn
signal
isConstant
foldable

findGlobals

```
findGlobals(fun, merge = TRUE)
```

```
fun  
merge
```

```
assignrm
```

```
mergefunctionsvariables
```

```
findGlobals(findGlobals)  
findGlobals(findGlobals, merge = FALSE)
```

showTree

```
showTree(e, write = cat)
```

```
e  
write
```

```
showTree(quote(-3))  
showTree(quote("x"<-1))  
showTree(quote("f"(x)))
```

foreign

lookup.xport

lookup.xport(file)

file

[https://support.sas.com/content/dam/SAS/support/en/technical-papers/
record-layout-of-a-sas-version-5-or-6-data-set-in-sas-transport-xport-format.
pdf](https://support.sas.com/content/dam/SAS/support/en/technical-papers/record-layout-of-a-sas-version-5-or-6-data-set-in-sas-transport-xport-format.pdf)

[read.xport](#)

```
## Not run: ## no XPORT file is installed.  
lookup.xport("test.xpt")
```

```
## End(Not run)
```

`read.arff`

`read.arff(file)`

`file` `connection`

https://waikato.github.io/weka-wiki/formats_and_processing/arff/

`write.arff``write.arff``read.arff`

`read.dbf`

`read.dbf(file, as.is = FALSE)`

`file`
`as.is`

<https://www.clicketyclick.dk/databases/xbase/format/>
`read.dbf`<http://shapelib.maptools.org/>"L""N""F""D"

`make.names(unique=TRUE)`
"data_type"

<http://shapelib.maptools.org/>

`write.dbf`

```
x <- read.dbf(system.file("files/sids.dbf", package="foreign"))[1]
str(x)
summary(x)
```

`read.dta`

```
read.dta(file, convert.dates = TRUE, convert.factors = TRUE,
         missing.type = FALSE,
         convert.underscore = FALSE, warn.missing.labels = TRUE)
```

```
file
convert.dates  DatePOSIXct
convert.factors
```

```
missing.type
convert.underscore
              " _ " . "
warn.missing.labels
```

`http:ftp:https:https:`

```
Dateread.dtaconvert.factors = NAconvert.dates = FALSE
missing.typeTRUENULLNA"missing"
format-115
```

```
"datalabel""time.stamp""formats""types""val.labels""var.labels""version"
"label.table""expansion.table"5, 6, 7-781012
"val.labels""label.table"
```

https://www.stata.com/help.cgi?dta_114https://www.stata.com/help.cgi?dta_113

Stata.fileread_dta

[write.dtaattributesDatefactor](#)

```
write.dta(swiss,swissfile <- tempfile())
read.dta(swissfile)
```

read.epiinfo

.REC

```
read.epiinfo(file, read.deleted = FALSE, guess.broken.dates = FALSE,
              thisyear = NULL, lower.case.names = FALSE)
```

file

```
read.deleted    TRUEFALSENANA
guess.broken.dates
```

thisyear

lower.case.names

Dateguess.broken.datesthisyear

read.deletedTRUE"deleted"

<https://www.cdc.gov/epiinfo/><http://www.epidata.dk>

Not run: ## That file is not available

```
read.epiinfo("oswego.rec", guess.broken.dates = TRUE, thisyear = "1972")
```

End(Not run)

read.mtp

read.mtp(file)

file

<https://www.minitab.com/>

```
## Not run:  
read.mtp("ex1-10.mtp")
```

```
## End(Not run)
```

read.octave

read.octave(file)

file

save -text *NULL*

<stephen@gnu.org>

<https://octave.org/>

read.spss

read.spsssaveexport

```
read.spss(file, use.value.labels = TRUE, to.data.frame = FALSE,
          max.value.labels = Inf, trim.factor.names = FALSE,
          trim_values = TRUE, reencode = NA, use.missings = to.data.frame,
          sub = ".", add.undeclared.levels = c("sort", "append", "no"),
          duplicated.value.labels = c("append", "condense"),
          duplicated.value.labels.infix = "_duplicated_", ...)

file
use.value.labels
      NA
to.data.frame
max.value.labels
      TRUE
trim.factor.names

trim_values      use.value.labels = TRUE
reencode         NA
use.missings     NA
sub              NAiconv"."sub=NA
add.undeclared.levels
      "sort""append""no"to.data.frame=TRUEadd.undeclared.levels="no"
      NA
duplicated.value.labels
      "append"paste0(label,    duplicated.value.labels.infix,    level)
      "condense"
duplicated.value.labels.infix
      "_duplicated_"duplicated.value.labels="append"
...              as.data.frame.to.data.frame = TRUE
```

<http://www.gnu.org/software/pspp/>

[http:ftp:https:https:download.filemethod](http://ftp://ftp://https://download.filemethod)

max.value.labels

"variable.labels"

trim_values=TRUEsub

<https://learn.microsoft.com/en-us/windows/win32/intl/code-page-identifiers>
[iconvreencodeiconvlist](#)

```
"codepage"
"label.table""variable.labels""label.table"NULL"variable.labels"
"Missings"typevalueread.spss"one""two""three""label.table"http://www.gnu.org/
software/pspp/manual/html\_node/Missing-Observations.html#Missing-Observations
```

1, 2, 3, ...

```
save"codepage"reencode
```

```
spss.system.file
```

```
(sav <- system.file("files", "electric.sav", package = "foreign"))
dat <- read.spss(file=sav)
str(dat) # list structure with attributes
```

```
dat <- read.spss(file=sav, to.data.frame=TRUE)
str(dat) # now a data.frame
```

```
### Now we use an example file that is not very well structured and
### hence may need some special treatment with appropriate argument settings.
### Expect lots of warnings as value labels (corresponding to R factor labels) are uncomplete,
### and an unsupported long string variable is present in the data
(sav <- system.file("files", "testdata.sav", package = "foreign"))
```

```
### Examples for add.undeclared.levels:
## add.undeclared.levels = "sort" (default):
x.sort <- read.spss(file=sav, to.data.frame = TRUE)
## add.undeclared.levels = "append":
x.append <- read.spss(file=sav, to.data.frame = TRUE,
  add.undeclared.levels = "append")
## add.undeclared.levels = "no":
x.no <- read.spss(file=sav, to.data.frame = TRUE,
  add.undeclared.levels = "no")
```

```
levels(x.sort$factor_n_undeclared)
levels(x.append$factor_n_undeclared)
str(x.no$factor_n_undeclared)
```

```
### Examples for duplicated.value.labels:
## duplicated.value.labels = "append" (default)
x.append <- read.spss(file=sav, to.data.frame=TRUE)
## duplicated.value.labels = "condense"
x.condense <- read.spss(file=sav, to.data.frame=TRUE,
  duplicated.value.labels = "condense")
```

```

levels(x.append$factor_n_duplicated)
levels(x.condense$factor_n_duplicated)

as.numeric(x.append$factor_n_duplicated)
as.numeric(x.condense$factor_n_duplicated)

## Long Strings (>255 chars) are imported in consecutive separate variables
## (see warning about subtype 14):
x <- read.spss(file=sav, to.data.frame=TRUE, stringsAsFactors=FALSE)

cat.long.string <- function(x, w=70) cat(paste(strwrap(x, width=w), "\n"))

## first part: x$string_500:
cat.long.string(x$string_500)
## second part: x$STRIN0:
cat.long.string(x$STRIN0)
## complete long string:
long.string <- apply(x[,c("string_500", "STRIN0")], 1, paste, collapse="")
cat.long.string(long.string)

```

read.ssd

read.xport

```

read.ssd(libname, sectionnames,
         tmpXport=tempfile(), tmpProgLoc=tempfile(), sascmd="sas")

```

```

libname
sectionnames    libname.ssd0x.sas7bdat
tmpXport
tmpProgLoc
sascmd

```

NULL

.CSV

<stvjc@channing.harvard.edu>

read.xport

```
## if there were some files on the web we could get a real
## runnable example
## Not run:
R> list.files("trialdata")
[1] "baseline.sas7bdat" "form11.sas7bdat" "form12.sas7bdat"
[4] "form13.sas7bdat" "form22.sas7bdat" "form23.sas7bdat"
[7] "form3.sas7bdat" "form4.sas7bdat" "form48.sas7bdat"
[10] "form50.sas7bdat" "form51.sas7bdat" "form71.sas7bdat"
[13] "form72.sas7bdat" "form8.sas7bdat" "form9.sas7bdat"
[16] "form90.sas7bdat" "form91.sas7bdat"
R> baseline <- read.ssd("trialdata", "baseline")
R> form90 <- read.ssd("trialdata", "form90")

## Or for a Windows example
sashome <- "/Program Files/SAS/SAS 9.1"
read.ssd(file.path(sashome, "core", "sashelp"), "retail",
         sascmd = file.path(sashome, "sas.exe"))

## End(Not run)
```

read.systat

read.systatSAVE*.sys*.syd

read.systat(file, to.data.frame = TRUE)

file

to.data.frame

mtype = 1

to.data.frameFALSE

"comment"


```
summary(iris)
iris.s <- read.systat(system.file("files/Iris.syd", package="foreign")[1])
str(iris.s)
summary(iris.s)
```

read.xport

```
read.xport(file, ...)
```

```
file
```

```
... as.data.frame
```

```
_.A.ZNA
```

```
 , , ,
```

```
<saikat@stat.wisc.edu>
```

```
https://support.sas.com/content/dam/SAS/support/en/technical-papers/  
record-layout-of-a-sas-version-5-or-6-data-set-in-sas-transport-xport-format.  
pdf
```

```
lookup.xport
```

```
## Not run: ## no XPORT file is installed  
read.xport("test.xpt")
```

```
## End(Not run)
```

S3 read functions

`data.dump`

```
data.restore(file, print = FALSE, verbose = FALSE, env = .GlobalEnv)
read.S(file)
```

`file` `data.dump`
`print`
`verbose`
`env`

```
read.S
data.restoredata.dumpdata.dump(..., oldStyle=TRUE)
```

```
read.S
data.restore
```

```
## if you have an S-PLUS _Data file containing 'myobj'
## Not run: read.S(file.path("_Data", "myobj"))
data.restore("dumpdata", print = TRUE)

## End(Not run)
```

`write.arff`

```
write.arff(x, file, eol = "\n", relation = deparse(substitute(x)))
```

`x`
`file` `""`
`eol`
`relation`

relationmake.namesX

https://waikato.github.io/weka-wiki/formats_and_processing/arff/

read.arffwrite.arffread.arff

write.arff(iris, file = "")

write.dbf

write.dbf(dataframe, file, factor2char = TRUE, max_nchar = 254)

dataframe

file

factor2char TRUE

max_nchar

"logical""numeric""integer""character""factor""Date"

max_nchar

NULL

"numeric""integer""character""Date""logical"0,-1

<http://shapelib.maptools.org/>

https://www.clicketyclick.dk/databases/xbase/format/data_types.html

[read.dbf](#)

```
str(warpbreaks)
try1 <- paste(tempfile(), ".dbf", sep = "")
write.dbf(warpbreaks, try1, factor2char = FALSE)
in1 <- read.dbf(try1)
str(in1)
try2 <- paste(tempfile(), ".dbf", sep = "")
write.dbf(warpbreaks, try2, factor2char = TRUE)
in2 <- read.dbf(try2)
str(in2)
unlink(c(try1, try2))
```

[write.dta](#)

[drop](#)

```
write.dta(dataframe, file, version = 7L,
          convert.dates = TRUE, tz = "GMT",
          convert.factors = c("labels", "string", "numeric", "codes"))
```

```
dataframe
file
version
convert.dates  DatePOSIXct
tz
convert.factors
```

[abbreviate](#)

```
convert.factors = "string"val.labelsconvert.factors = "numeric"NA
convert.factors = "codes"
"label.table"val.labels"
"datalabel""Written by R."
"expansion.table"listcharacter
"val.labels"character
"var.labels"character
version = 7
```

21474836208.988e+307

NULL

```
convert.dates = FALSE
```

```
format xdate %td;
```

```
POSIXct%tc%tCPOSIXctconvert.dates = FALSE3156192001000write.dta%tchttps://www.stata.com/manuals13/ddatetime.pdf
```

https://www.stata.com/help.cgi?dta_114https://www.stata.com/help.cgi?dta_113

[read.dtaattributesDateTimeClassesabbreviate](#)

```
write.dta(swiss, swissfile <- tempfile())
read.dta(swissfile)
```

write.foreign

```
write.foreign(df, datafile, codefile,
              package = c("SPSS", "Stata", "SAS"), ...)
```

```
df
datafile
codefile
package
...          writeForeign
```

```
foreign::writeForeignStataforeign::writeForeignSASforeign::writeForeignSPSS
writeForeignSystatwrite.foreignwrite.foreignpackage="Systat"
```

```
DatedatesdatePOSIXt
```

```
package="SAS"dataname = "rdata"validvarname"V6""V7"libpath = NULL
```

```
package="SPSS"maxchars = 32Lmaxcharsmaxchars = 8Lmaxchars = 64L
```

```
package="SPSS"SET DECIMAL=DOT.
```

NULL

```
## Not run:
datafile <- tempfile()
codefile <- tempfile()
write.foreign(esoph, datafile, codefile, package="SPSS")
file.show(datafile)
file.show(codefile)
unlink(datafile)
unlink(codefile)

## End(Not run)
```


lattice

A.01 Lattice

```
xyplot  
trellis.par.setlattice.optionstrellis.device  
print.trellisupdate.trellis"trellis"locatortrellis.focus  
https://cran.r-project.org/package=lattice
```

```
xyplot"trellis"printplotsourceforwhileprintplot  
par()
```

<Deepayan.Sarkar@R-project.org>

<http://lmdvr.r-forge.r-project.org/>

<http://web.archive.org/web/20081020164041/http://cm.bell-labs.com/cm/ms/departments/sia/project/trellis/display.writing.html>


```
panel.xyplotpanel.xylothistogrampanel.histogram
```

```
barchart
```

```
bwplot
```

```
densityplot
```

```
dotplot
```

```
histogram
```

```
qqmath
```

```
stripplot
```

```
qq
```

```
xyplot
```

```
levelplot image
```

```
contourplot
```

```
cloud
```

```
wireframe persp
```

```
splom
```

```
parallel
```

```
rfs oneway
```

```
tmd
```

```
panel.functionslinespointsllinespanel.lines
```

```
## Not run:
```

```
## Show brief history of changes to lattice, including  
## a summary of new features.
```

```
RShowDoc("NEWS", package = "lattice")
```

```
## End(Not run)
```

xyplotbwplotdotplotbarchartstripplot

```
xyplot(x, data, ...)
dotplot(x, data, ...)
barchart(x, data, ...)
stripplot(x, data, ...)
bwplot(x, data, ...)
```

```
## S3 method for class 'formula'
```

```
xyplot(x,
      data,
      allow.multiple = is.null(groups) || outer,
      outer = !is.null(groups),
      auto.key = lattice.getOption("default.args")$auto.key,
      aspect = "fill",
      panel = lattice.getOption("panel.xyplot"),
      prepanel = NULL,
      scales = list(),
      strip = TRUE,
      groups = NULL,
      xlab,
      xlim,
      ylab,
      ylim,
      drop.unused.levels = lattice.getOption("drop.unused.levels"),
      ...,
      lattice.options = NULL,
      default.scales,
      default.prepanel = lattice.getOption("prepanel.default.xyplot"),
      subscripts = !is.null(groups),
      subset = TRUE)
```

```
## S3 method for class 'data.frame'
```

```
xyplot(x, data = NULL, formula = data, ...)
```

```
## S3 method for class 'formula'
```

```
dotplot(x,
      data,
      panel = lattice.getOption("panel.dotplot"),
      default.prepanel = lattice.getOption("prepanel.default.dotplot"),
      ...)
```

```
## S3 method for class 'data.frame'
```

```
dotplot(x, data = NULL, formula = data, ...)
```

```

## S3 method for class 'formula'
barchart(x,
  data,
  panel = lattice.getOption("panel.barchart"),
  default.prepanel = lattice.getOption("prepanel.default.barchart"),
  box.ratio = 2,
  ...)

## S3 method for class 'data.frame'
barchart(x, data = NULL, formula = data, ...)

## S3 method for class 'formula'
stripplot(x,
  data,
  panel = lattice.getOption("panel.stripplot"),
  default.prepanel = lattice.getOption("prepanel.default.stripplot"),
  ...)

## S3 method for class 'data.frame'
stripplot(x, data = NULL, formula = data, ...)

## S3 method for class 'formula'
bwplot(x,
  data,
  allow.multiple = is.null(groups) || outer,
  outer = FALSE,
  auto.key = lattice.getOption("default.args")$auto.key,
  aspect = "fill",
  panel = lattice.getOption("panel.bwplot"),
  prepanel = NULL,
  scales = list(),
  strip = TRUE,
  groups = NULL,
  xlab,
  xlim,
  ylab,
  ylim,
  box.ratio = 1,
  horizontal = NULL,
  drop.unused.levels = lattice.getOption("drop.unused.levels"),
  ...,
  lattice.options = NULL,
  default.scales,
  default.prepanel = lattice.getOption("prepanel.default.bwplot"),
  subscripts = !is.null(groups),
  subset = TRUE)

## S3 method for class 'data.frame'
bwplot(x, data = NULL, formula = data, ...)

```

```

"formula"x
y ~ x | g1 * g2 * ...y ~ x | g1 + g2 + ...yxg1, g2, ...xyg1, g2,
...y ~ xsqrt()log()data
xyplot ~ x | g1 * g2 * ...ynames(x)x
xnumericbarchartdotplotbarchart.table
g1, g2, ...shingle"levels"shingleequal.count
yxy1 + y2 ~ x | a * by1 ~ x | a * by2 ~ x | a * by1 ~ xy2 ~ x
groupsy1y2groupsreshapegroups
y1 + y2allow.multiple=FALSEI(y1+y2)
outerTRUE
xyxyplotxyhorizontalhorizontal=TRUEyxhorizontalFALSExTRUE
horizontal=FALSE
xformula
data      formulaenvirevalgroupssubsetdatadataxdata
formula   "data.frame"x
allow.multiple TRUE
outer     xFALSEgroups
box.ratio barchartbwplotbox.width
horizontal bwplotdotplotbarchartstriplotxyxFALSExTRUE
scalesabbreviate = TRUEscalesminlengthabbreviate

panel     xypanelpanel.panel.xyplotpanel.barchart
xyxyplot...
xy
panel.gridpanel.loesslinesllinespanel.lineslines
groupsNULLgroupssubscriptspanel.superposepanel.groupsgroups
xyplotbarchartdotplotstriplot
groupssubscriptssubscriptscopydatasubsetgroupsgroups[subscripts]
groups
subscriptssallow.multiplelatticParseFormulasubscriptssubset
subscripts
panel.numberpacket.numbertrellis.focusperm.condindex.cond
packet.number
panel.xyplottypexyplotgroupspanel.xyplotpanel.superposexyplot
bwplothorizontalx
aspect    "fill""xy"banking"iso"
prepanelldxy
groups    datagroupsgroupssubscripts
keyauto.keysimpleKey
auto.key  simpleKeylattic.options
auto.key = TRUEauto.key = list()simpleKeykeysimpleKey
auto.keyauto.key = TRUEgroupsauto.keyauto.key=list(columns = 2)
columnskey
auto.keyFALSEgroupskeylegendsimpleKeylevels(groups)textauto.key
textgroups
simpleKeytrellis.par.getpar.settingssimpleTheme
keysimpleKey"trellis"plotauto.key

```

```

prepanel      panelxlimylimdxdyxatyat
              xlimylimxlimylimxlimlimscalesrelationscales"same"
              dxdyaspect"xy"banking
              xlimylimnxlimylimnxlimylim
              xatyatprepanel1:nnrelation="free"relation="sliced"scales
              xyxlimylim

strip         FALSEstripstrip.defaultstrip.defaultstrip.left...
xlab          "grob"xNULLmainlabelxlab
ylab          "grob"ymainxlab
scales        name=valuexyxyscalesxybwplotalternating=FALSEscales
              scalesxyrelation

              relation "same""free""sliced"relation="same"relation="free"
              relation="sliced"
              prepanelxlimlimscales$limitsrelation="sliced"relation
              "free"xlimylim
              tick.number xlim
              draw TRUE
              alternating alternating

              alternatingrelation="same"TRUEc(1, 2)
              limits xlimylim
              at
              labels atat
              cex
              fontfontfacefontfamily
              lineheight cloud
              tck
              col
              rot
              abbreviate abbreviate
              minlength abbreviateabbreviate=TRUE
              log xyFALSETRUE"e"scales=list(x = list(log = 2))y ~ log2(x)
              equispaced.log
              equispaced.log TRUE
              equispaced.logFALSETRUE"base^loc"baseloc
              xscale.components.logpower
              format formatstrptime
              axs "r""i"

              scalespscalessplom

subscripts    subscriptsFALSEgroupssubscriptspanel.identify
subset        groupsdatadatassubscriptsTRUEdatadrop.unused.levels

```

```

xlim          x1:length(xlim)
              xlimscales$x$relation"free"

ylim          xlim
drop.unused.levels
              interactionconddata$attice.getOption("drop.unused.levels")TRUE
              groups

default.scales scales
default.prepanel
              prepanel$attice.options

lattice.options
              $attice.options$par.settings

...

trellis.skeletonpanel.bwplotpanel.bwplot

as.table FALSETRUE
between xyxybetween
key draw.key"grobgrob""trellis"
  simpleKeyauto.keykeylegend
  spacespace"top""bottom""left""right"xycornerxycornerc(0,0)
  c(1,0)c(1,1)c(0,1)xy
  xy"legend.bbox"$attice.getOption"full""panel"
  "rectangles""lines""points""text"key
  "text"rep"text"
  key"text""lines""points""rectangles"
    cex=1
    col="black"
    alpha=1
    fill="transparent"
    lty=1
    lwd=1
    font=1
    fontface
    fontfamily
    pch=8
    adj=0
    type="l"
    size=5
    height=1
    lineheight=1
    angle=0
    density=-1
  border"rect"border
  angledensitysizetype"l""p""b""o"height
  key
reverse.rows FALSETRUEerepbarchartstack = TRUEreverse.rows =
  TRUE
between

```

```

title
rep TRUE"text"rep=FALSE
cex.title
lines.title
padding.text 0.2 * "lines"
background
alpha.background
border borderTRUEFALSE
transparent=FALSE
just grid.layout
columns
between.columns between
divide type"b""o"lines
legend "grob"
  legend"left""right""top""bottom""inside""inside"spacekeykey
  colorkeylevelplotwireframespacelegend
  legendfun"grob""grob"args"inside"xycornerkey
page
xlab.topylab.right xlabylab
main NULL
  mainxlabylabsubplotmathylab
  mainlabelxlabylabmainsubtextGrobjustrotcolfontgpar
  mainsubxlabylabxlab.topylab.right"grob"
sub "grob"main
par.strip.text colcexfontlines"\n"
  lineheightabbreviate = TRUEminlengthdotabbreviate
layout layout
  layoutc(0,n)npar("din")NA

skip FALSETRUE
strip.left strip.leftstrip
xlab.defaultylab.default xlabylabNULL
xscale.componentsyscale.components xscale.components.default
axis axis.default
perm.cond 1:nnperm.condnperm.condpermaperm
index.cond perm.condindex.condindex.cond
  index.condindex.condilevels(g_i)g_iisubset
  drop.unused.levelsTRUE
  index.condperm.cond
  index.cond...
  perm.condindex.cond"trellis"update.trellisNULL
par.settings trellis.par.setlattice.options
plot.args plot.trellisplotprint

```

```

formulag1, g2, ...(x,y)layoutindex.cond
as.table = TRUEas.table=TRUElattice.options(default.args = list(as.table = TRUE))
as.tableaspectbetweenpagemainsubpar.strip.textlayoutskskipstrip
g1g2index.condperm.condupdate

```

"trellis"updateprint

<Deepayan.Sarkar@R-project.org>

<http://lmdvr.r-forge.r-project.org/>

Latticebarchart.tableprint.trellisshinglebankingreshapepanel.xyplotpanel.bwplot
panel.barchartpanel.dotplotpanel.stripplotpanel.superposepanel.loess
panel.averagestrip.defaultsimpleKeytrellis.par.set

```
require(stats)

## Tonga Trench Earthquakes

Depth <- equal.count(quakes$depth, number=8, overlap=.1)
xyplot(lat ~ long | Depth, data = quakes)
update(trellis.last.object(),
       strip = strip.custom(strip.names = TRUE, strip.levels = TRUE),
       par.strip.text = list(cex = 0.75),
       aspect = "iso")

## Extended formula interface

xyplot(Sepal.Length + Sepal.Width ~ Petal.Length + Petal.Width | Species,
       data = iris, scales = "free", layout = c(2, 2),
       auto.key = list(x = .75, y = .75, corner = c(0.5, 0.5)))

## user defined panel functions

states <- data.frame(state.x77,
                     state.name = dimnames(state.x77)[[1]],
                     state.region = state.region)
xyplot(Murder ~ Population | state.region, data = states,
       snames = states$state.name,
       panel = function(x, y, subscripts, snames) {
         panel.text(x = x, y = y, labels = snames[subscripts], cex = 1,
                   fontfamily = "HersheySans")
       })

## Stacked bar chart

barchart(yield ~ variety | site, data = barley,
         groups = year, layout = c(1,6), stack = TRUE,
         auto.key = list(space = "right"),
```



```

        ylab = "Barley Yield (bushels/acre)",
        scales = list(x = list(rot = 45)))

bwplot(voice.part ~ height, data = singer, xlab = "Height (inches)")

dotplot(variety ~ yield | year * site, data=barley)

## Grouped dot plot showing anomaly at Morris

dotplot(variety ~ yield | site, data = barley, groups = year,
        key = simpleKey(levels(barley$year), space = "right"),
        xlab = "Barley Yield (bushels/acre) ",
        aspect=0.5, layout = c(1,6), ylab=NULL)

stripplot(voice.part ~ jitter(height), data = singer, aspect = 1,
          jitter.data = TRUE, xlab = "Height (inches)")

## Interaction Plot

xyplot(decrease ~ treatment, OrchardSprays, groups = rowpos,
       type = "a",
       auto.key =
       list(space = "right", points = FALSE, lines = TRUE))

## longer version with no x-ticks

## Not run:
bwplot(decrease ~ treatment, OrchardSprays, groups = rowpos,
       panel = "panel.superpose",
       panel.groups = "panel.linejoin",
       xlab = "treatment",
       key = list(lines = Rows(trellis.par.get("superpose.line"),
                             c(1:7, 1)),
                 text = list(lab = as.character(unique(OrchardSprays$rowpos))),
                 columns = 4, title = "Row position"))

## End(Not run)

```

B.01 xyplot.ts

```

## S3 method for class 'ts'
xyplot(x, data = NULL,
       screens = if (superpose) 1 else colnames(x),
       ...,
       superpose = FALSE,
       cut = FALSE,
       type = "l",
       col = NULL,
       lty = NULL,

```

```

lwd = NULL,
pch = NULL,
cex = NULL,
fill = NULL,
auto.key = superpose,
panel = if (superpose) "panel.superpose"
        else "panel.superpose.plain",
par.settings = list(),
layout = NULL, as.table = TRUE,
xlab = "Time", ylab = NULL,
default.scales = list(y = list(relation =
    if (missing(cut)) "free" else "same")))

```

```

x          ts
data       NULL
...        xyplotpanel.xyplot
screens    screens = c(1, 2, 1)
superpose  screens = 1trellis.par.get("superpose.line")
cut        cutlistequal.countnumberoverlapcutnumber
           cut = TRUEbankingaspect = "xy"
typecolltylwdpchcexfill
           panel.xyplot
auto.key   auto.keyxyplot
panel      panel.groupspanel.superpose
par.settings colltylwdtrellis.par.setsimpleTheme
layout
as.table   x
xlabylab   xyplotylabas.tableFALSE
default.scales scales"free"cutscalesdefault.scales

```

```
list(A = c(1,2), c(3,4))c(3, 4)c(1, 2)Ascreens
```

```
"trellis"updateprint
```

```
<felix@nfrac.org>
```

```
xyplot.tsscreensxyplot.ts
```

```
http://lmdvr.r-forge.r-project.org/
```

```
xyplotpanel.xyplotplot.tststxyplot.zoo
```

```

xyplot(ts(c(1:10,10:1)))

### Figure 14.1 from Sarkar (2008)
xyplot(sunspot.year, aspect = "xy",
       strip = FALSE, strip.left = TRUE,
       cut = list(number = 4, overlap = 0.05))

### A multivariate example; first juxtaposed, then superposed
xyplot(EuStockMarkets, scales = list(y = "same"))
xyplot(EuStockMarkets, superpose = TRUE, aspect = "xy", lwd = 2,
       type = c("l", "g"), ylim = c(0, max(EuStockMarkets)))

### Examples using screens (these two are identical)
xyplot(EuStockMarkets, screens = c(rep("Continental", 3), "UK"))
xyplot(EuStockMarkets, screens = list(FTSE = "UK", "Continental"))

### Automatic group styles
xyplot(EuStockMarkets, screens = list(FTSE = "UK", "Continental"),
       superpose = TRUE)

xyplot(EuStockMarkets, screens = list(FTSE = "UK", "Continental"),
       superpose = TRUE, xlim = extendrange(1996:1998),
       par.settings = standard.theme(color = FALSE))

### Specifying styles for series by name
xyplot(EuStockMarkets, screens = list(FTSE = "UK", "Continental"),
       col = list(DAX = "red", FTSE = "blue", "black"), auto.key = TRUE)

xyplot(EuStockMarkets, screens = list(FTSE = "UK", "Continental"),
       col = list(DAX = "red"), lty = list(SMI = 2), lwd = 1:2,
       auto.key = TRUE)

### Example with simpler data, few data points
set.seed(1)
z <- ts(cbind(a = 1:5, b = 11:15, c = 21:25) + rnorm(5))
xyplot(z, screens = 1)
xyplot(z, screens = list(a = "primary (a)", "other (b & c)"),
       type = list(a = c("p", "h"), b = c("p", "s"), "o"),
       pch = list(a = 2, c = 3), auto.key = list(type = "o"))

```

B.02 barchart.table

```

## S3 method for class 'table'
barchart(x, data, groups = TRUE,
        origin = 0, stack = TRUE, ..., horizontal = TRUE)

## S3 method for class 'array'
barchart(x, data, ...)

```

```
## S3 method for class 'matrix'
barchart(x, data, ...)

## S3 method for class 'table'
dotplot(x, data, groups = TRUE, ..., horizontal = TRUE)

## S3 method for class 'array'
dotplot(x, data, ...)

## S3 method for class 'matrix'
dotplot(x, data, ...)
```

```
x          tablearraymatrix
data
groups
originstack  panel.barcharttable
horizontal
...          formula
```

[tapermas.data.frame](#)

"trellis"[updateprint](#)

<Deepayan.Sarkar@R-project.org>

[barcharttapermtablepanel.barchartLattice](#)

```
barchart(Titanic, scales = list(x = "free"),
         auto.key = list(title = "Survived"))
```

```

histogram(x, data, ...)
densityplot(x, data, ...)

## S3 method for class 'formula'
histogram(x,
  data,
  allow.multiple, outer = TRUE,
  auto.key = lattice.getOption("default.args")$auto.key,
  aspect = "fill",
  panel = lattice.getOption("panel.histogram"),
  prepanel, scales, strip, groups,
  xlab, xlim, ylab, ylim,
  type = c("percent", "count", "density"),
  nint = if (is.factor(x)) nlevels(x)
  else round(log2(length(x)) + 1),
  endpoints = extend.limits(range(as.numeric(x),
    finite = TRUE), prop = 0.04),
  breaks,
  equal.widths = TRUE,
  drop.unused.levels =
    lattice.getOption("drop.unused.levels"),
  ...,
  lattice.options = NULL,
  default.scales = list(),
  default.prepanel =
    lattice.getOption("prepanel.default.histogram"),
  subscripts,
  subset)

## S3 method for class 'data.frame'
histogram(x, data = NULL, formula = data, ...)

## S3 method for class 'numeric'
histogram(x, data = NULL, xlab, ...)

## S3 method for class 'factor'
histogram(x, data = NULL, xlab, ...)

## S3 method for class 'formula'
densityplot(x,
  data,
  allow.multiple = is.null(groups) || outer,
  outer = !is.null(groups),
  auto.key = lattice.getOption("default.args")$auto.key,
  aspect = "fill",
  panel = lattice.getOption("panel.densityplot"),
  prepanel, scales, strip, groups, weights,
  xlab, xlim, ylab, ylim,
  bw, adjust, kernel, window, width, give.Rkern,
  n = 512, from, to, cut, na.rm,
  drop.unused.levels =

```

```

        lattice.getOption("drop.unused.levels"),
        ...,
        lattice.options = NULL,
        default.scales = list(),
        default.prepanel =
            lattice.getOption("prepanel.default.densityplot"),
        subscripts,
        subset)

## S3 method for class 'data.frame'
densityplot(x, data = NULL, formula = data, ...)

## S3 method for class 'numeric'
densityplot(x, data = NULL, xlab, ...)

do.breaks(endpoints, nint)

x
        formulax~ x | g1 * g2 * ...xg1g2xhistogramg1g2
        ~ x1 + x2 | g1 * g2xyplot
        numericfactorx
data        formulaxyplotdata
formula     "data.frame"x
type        "percent""count""density"
            type"density"breaksNULL"percent"
nint        breaksNULL
endpoints   breaksdo.breaks
breaks      type
            breakslattice.getOption("histogram.breaks")NULL
            breaks = seq_len(1 + nlevels(x)) - 0.5

x
            breaks = do.breaks(endpoints, nint)

NULLbreaksbreakshistbreaksnint
breaksNULLnintequal.widths
equal.widths breaks=NULLTRUE
n            density
panel       panel.histogrampanel.densityplot
allow.multipleouter
            xyplot
auto.key    xyplot
aspect     xyplot
prepanel   xyplot

```

scales	xyplot
strip	xyplot
groups	xyplot histogram densityplot
xlaby	lab
xlimylim	xyplot
drop.unused.levels	xyplot
lattice.options	xyplot
default.scales	xyplot
subscripts	xyplot
subset	xyplot
default.prepanel	xyplot
weights	groupssubscriptsx weights
bwadjustwidth	density
kernelwindow	density
give.Rkern	density TRUE
fromtocut	density
na.rm	NA density density TRUE
...	xyplot

[histogram](#)[densityplot](#)[density](#)[density](#)[densityplot](#)[density](#)
[xyplot](#)
[do.breaks](#)

"trellis"[update](#)[print](#)

[panel.histogram](#)[histogramx](#)

<Deepayan.Sarkar@R-project.org>

<http://lmdvr.r-forge.r-project.org/>

[xyplot](#)[panel.histogram](#)[densitypanel.densityplot](#)[panel.mathdensity](#)[Lattice](#)

```

require(stats)
histogram( ~ height | voice.part, data = singer, nint = 17,
           endpoints = c(59.5, 76.5), layout = c(2,4), aspect = 1,
           xlab = "Height (inches)")

histogram( ~ height | voice.part, data = singer,
           xlab = "Height (inches)", type = "density",
           panel = function(x, ...) {
             panel.histogram(x, ...)
             panel.mathdensity(dmath = dnorm, col = "black",
                               args = list(mean=mean(x),sd=sd(x)))
           } )

densityplot( ~ height | voice.part, data = singer, layout = c(2, 4),
             xlab = "Height (inches)", bw = 5)

```

B.04 qqmath

```

qqmath(x, data, ...)

## S3 method for class 'formula'
qqmath(x,
       data,
       allow.multiple = is.null(groups) || outer,
       outer = !is.null(groups),
       distribution = qnorm,
       f.value = NULL,
       auto.key = lattice.getOption("default.args")$auto.key,
       aspect = "fill",
       panel = lattice.getOption("panel.qqmath"),
       prepanel = NULL,
       scales, strip, groups,
       xlab, xlim, ylab, ylim,
       drop.unused.levels = lattice.getOption("drop.unused.levels"),
       ...,
       lattice.options = NULL,
       default.scales = list(),
       default.prepanel = lattice.getOption("prepanel.default.qqmath"),
       subscripts,
       subset)

## S3 method for class 'data.frame'
qqmath(x, data = NULL, formula = data, ...)

## S3 method for class 'numeric'
qqmath(x, data = NULL, ylab, ...)

```



```

x
      "formula"x~ x | g1 * g2 * ...x"numeric"x
data      formulaxyplotdata
formula    "data.frame"x
distribution  qnormqunifqt
f.value    ppointsquantiledistribution
           f.valueNULLppointsqqnormf.value=ppoints
           xtails.npanel.qqmath
panel      panel.qqmath
allow.multipleouter
           xyplot
auto.key    xyplot
aspect      xyplot
prepanel    xyplot
scales      xyplot
strip       xyplot
groups      xyplot
xlabylab    xyplot
xlimylim    xyplot
drop.unused.levels
           xyplot
lattice.options
           xyplot
default.scales xyplot
subscripts    xyplot
subset         xyplot
default.prepanel
           xyplot
...           xyplot

```

```

qqmathqqmathqqnormf.value
qqmathdistributionf.valuepanel.qqmath
xyplot

```

```

"trellis"updateprint

```

```

<Deepayan.Sarkar@R-project.org>

```

```

xyplotpanel.qqmathpanel.qqmathlineprepanel.qqmathlineLatticequantile

```

```

qqmath(~ rnorm(100), distribution = function(p) qt(p, df = 10))
qqmath(~ height | voice.part, aspect = "xy", data = singer,
       prepanel = prepanel.qqmathline,
       panel = function(x, ...) {
         panel.qqmathline(x, ...)
         panel.qqmath(x, ...)
       })
vp.comb <-
  factor(sapply(strsplit(as.character(singer$voice.part), split = " "),
               "[", 1),
         levels = c("Bass", "Tenor", "Alto", "Soprano"))
vp.group <-
  factor(sapply(strsplit(as.character(singer$voice.part), split = " "),
               "[", 2))
qqmath(~ height | vp.comb, data = singer,
       groups = vp.group, auto.key = list(space = "right"),
       aspect = "xy",
       prepanel = prepanel.qqmathline,
       panel = function(x, ...) {
         panel.qqmathline(x, ...)
         panel.qqmath(x, ...)
       })

```

B.05 qq

```

qq(x, data, ...)

## S3 method for class 'formula'
qq(x, data, aspect = "fill",
   panel = lattice.getOption("panel.qq"),
   prepanel, scales, strip,
   groups, xlab, xlim, ylab, ylim, f.value = NULL,
   drop.unused.levels = lattice.getOption("drop.unused.levels"),
   ...,
   lattice.options = NULL,
   qtype = 7,
   default.scales = list(),
   default.prepanel = lattice.getOption("prepanel.default.qq"),
   subscripts,
   subset)

## S3 method for class 'data.frame'
qq(x, data = NULL, formula = data, ...)

```

```

x
"formula"xy ~ x | g1 * g2 * ...xyyx
data      formulaxyplot
formula   "data.frame"x
f.value   ppointsquantile
           f.valueNULL
           f.value = function(n) ppoints(n, a = 1)

```

```

           qqplotqq
           x
panel      panel.qq
qtype      typequantile
aspect     xyplot
prepanel   xyplot
scales     xyplot
strip      xyplot
groups     xyplot
xlabylab   xyplot
xlimylim   xyplot
drop.unused.levels
           xyplot
lattice.options
           xyplot
default.scales xyplot
subscripts    xyplot
subset        xyplot
default.prepanel
           xyplot
...           xyplot

```

```

qqqqf.value
xyplot

```

```

"trellis"updateprint

```

```

<Deepayan.Sarkar@R-project.org>

```

```

xyplotpanel.qqqmathLattice

```

```

qq(voice.part ~ height, aspect = 1, data = singer,
   subset = (voice.part == "Bass 2" | voice.part == "Tenor 1"))

```

B.06 levelplot

```
levelplot(x, data, ...)
contourplot(x, data, ...)

## S3 method for class 'formula'
levelplot(x,
  data,
  allow.multiple = is.null(groups) || outer,
  outer = TRUE,
  aspect = "fill",
  panel = if (useRaster) lattice.getOption("panel.levelplot.raster")
    else lattice.getOption("panel.levelplot"),
  prepanel = NULL,
  scales = list(),
  strip = TRUE,
  groups = NULL,
  xlab,
  xlim,
  ylab,
  ylim,
  at,
  cuts = 15,
  pretty = FALSE,
  region = TRUE,
  drop.unused.levels =
    lattice.getOption("drop.unused.levels"),
  ...,
  useRaster = FALSE,
  lattice.options = NULL,
  default.scales = list(),
  default.prepanel =
    lattice.getOption("prepanel.default.levelplot"),
  colorkey = region,
  col.regions,
  alpha.regions,
  subset = TRUE)

## S3 method for class 'formula'
contourplot(x,
  data,
  panel = lattice.getOption("panel.contourplot"),
  default.prepanel =
    lattice.getOption("prepanel.default.contourplot"),
  cuts = 7,
  labels = TRUE,
```

```

        contour = TRUE,
        pretty = TRUE,
        region = FALSE,
        ...)

## S3 method for class 'data.frame'
levelplot(x, data = NULL, formula = data, ...)

## S3 method for class 'data.frame'
contourplot(x, data = NULL, formula = data, ...)

## S3 method for class 'table'
levelplot(x, data = NULL, aspect = "iso", ..., xlim, ylim)

## S3 method for class 'table'
contourplot(x, data = NULL, aspect = "iso", ..., xlim, ylim)

## S3 method for class 'matrix'
levelplot(x, data = NULL, aspect = "iso",
          ..., xlim, ylim,
          row.values = seq_len(nrow(x)),
          column.values = seq_len(ncol(x)))

## S3 method for class 'matrix'
contourplot(x, data = NULL, aspect = "iso",
            ..., xlim, ylim,
            row.values = seq_len(nrow(x)),
            column.values = seq_len(ncol(x)))

## S3 method for class 'array'
levelplot(x, data = NULL, ...)

## S3 method for class 'array'
contourplot(x, data = NULL, ...)

x          formulaz ~ x * y | g1 * g2 * ...zxyg1, g2, ...

          levelplotwireframematrixarraytablexzyfilled.contourimage
          make.unique

data       formulagroupssubset
formula    "data.frame"x
row.valuescolumn.values
          xrow.valuescolumn.valuesnrow(x)ncol(x)

panel      xyplot
aspect     matrixaspect="fill"xyplot
at         zcol.regionsatzlimatz
          atregion=FALSE

```

```

col.regions      col.regionslevel.colors
alpha.regions
colorkey

      space "left""right""top""bottom""right"
      xy
      col col.regionslevel.colors
      at
      tri.lowertri.upper NAat-InfInfTRUE
      labels atlabelsatcexcolrotfontfontfacefontfamily
      title "grob"mainxyplotlabeltextGrob
            title.controlrotitle.control$side
            titleNULL
      title.control side"top""bottom""left""right""space"padding
      tick.number
      tck
      corner
      width
      height
      raster grid.rasterpanel.levelplot.raster
      interpolate rasterGrobraster=TRUE
      axis.line trellis.par.get("axis.line")
      axis.text trellis.par.get("axis.text")

contour
cuts      z
labels      panel.levelplotlabel.style
pretty
region
allow.multipleouterprepanelscalesstripgroupsxlabxlimylabylimdrop.unused.levels
lattice.optionsdefault.scalessubset
      xyplot
default.prepanel
      xyplot
...      levelplotcontourplot
useRaster TRUEpanel.levelplotpanel.levelplot.rastercolorkey$rasterTRUE
      panel.levelplot.rasterpanel.levelplotuseRaster=TRUE
      useRaster=TRUE

xyplot
panel.levelplot

"trellis"updateprint

```

<Deepayan.Sarkar@R-project.org>

<http://lmdvr.r-forge.r-project.org/>

`xyplotLatticepanel.levelplot`

```
x <- seq(pi/4, 5 * pi, length.out = 100)
y <- seq(pi/4, 5 * pi, length.out = 100)
r <- as.vector(sqrt(outer(x^2, y^2, "+")))
grid <- expand.grid(x=x, y=y)
grid$z <- cos(r^2) * exp(-r/(pi^3))
levelplot(z ~ x * y, grid, cuts = 50, scales=list(log="e"), xlab="",
          ylab="", main="Weird Function", sub="with log scales",
          colorkey = FALSE, region = TRUE)
## triangular end-points in color key, with a title
levelplot(z ~ x * y, grid, col.regions = hcl.colors(10),
          at = c(-Inf, seq(-0.8, 0.8, by = 0.2), Inf))

#S-PLUS example
require(stats)
attach(environmental)
ozo.m <- loess((ozone^(1/3)) ~ wind * temperature * radiation,
              parametric = c("radiation", "wind"), span = 1, degree = 2)
w.marginal <- seq(min(wind), max(wind), length.out = 50)
t.marginal <- seq(min(temperature), max(temperature), length.out = 50)
r.marginal <- seq(min(radiation), max(radiation), length.out = 4)
wtr.marginal <- list(wind = w.marginal, temperature = t.marginal,
                    radiation = r.marginal)
grid <- expand.grid(wtr.marginal)
grid[, "fit"] <- c(predict(ozo.m, grid))
contourplot(fit ~ wind * temperature | radiation, data = grid,
            cuts = 10, region = TRUE,
            xlab = "Wind Speed (mph)",
            ylab = "Temperature (F)",
            main = "Cube Root Ozone (cube root ppb)")
detach()
```

B.07 cloud

"formula"

```
cloud(x, data, ...)
wireframe(x, data, ...)
```

```
## S3 method for class 'formula'
```

```
cloud(x,
      data,
      allow.multiple = is.null(groups) || outer,
      outer = FALSE,
```

```

auto.key = lattice.getOption("default.args")$auto.key,
aspect = c(1,1),
panel.aspect = 1,
panel = lattice.getOption("panel.cloud"),
prepanel = NULL,
scales = list(),
strip = TRUE,
groups = NULL,
xlab,
ylab,
zlab,
xlim = if (is.factor(x)) levels(x) else range(x, finite = TRUE),
ylim = if (is.factor(y)) levels(y) else range(y, finite = TRUE),
zlim = if (is.factor(z)) levels(z) else range(z, finite = TRUE),
at,
drape = FALSE,
pretty = FALSE,
drop.unused.levels,
...,
lattice.options = NULL,
default.scales =
list(distance = c(1, 1, 1),
      arrows = TRUE,
      axs = axs.default),
default.prepanel = lattice.getOption("prepanel.default.cloud"),
colorkey,
col.regions,
alpha.regions,
cuts = 70,
subset = TRUE,
axs.default = "r")

## S3 method for class 'data.frame'
cloud(x, data = NULL, formula = data, ...)

## S3 method for class 'formula'
wireframe(x,
          data,
          panel = lattice.getOption("panel.wireframe"),
          default.prepanel = lattice.getOption("prepanel.default.wireframe"),
          ...)

## S3 method for class 'data.frame'
wireframe(x, data = NULL, formula = data, ...)

## S3 method for class 'matrix'
cloud(x, data = NULL, type = "h",
      zlab = deparse(substitute(x)), aspect, ...,
      xlim, ylim, row.values, column.values)

## S3 method for class 'table'
cloud(x, data = NULL, groups = FALSE,

```



```

      zlab = deparse(substitute(x)),
      type = "h", ...)

## S3 method for class 'matrix'
wireframe(x, data = NULL,
          zlab = deparse(substitute(x)), aspect, ...,
          xlim, ylim, row.values, column.values)

x
      "formula"z ~ x * y | g1 * g2 * ...zxyg1, g2, ...wireframexy
      wireframexyz
      NAzNAz
      wireframecloudmatrixxxzxyersp
data      "formula"groupssubsetdata"formula"
formula    "data.frame"x
row.valuescolumn.values
      xrow.valuescolumn.valuesnrow(x)ncol(x)
allow.multipleouterauto.keyprepanelstripgroupsxlabxlimylabylimdrop.unused.levels
lattice.optionsdefault.scalesssubset
      xyplotcloud.tablegroups
type      cloudpanel.3dscatter"h"matrix
aspectpanel.aspect
      aspectaspectxyplotpanel.aspect
      matrixncol(x) / nrow(x)
panel      panel.cloud
default.prepanel
      xyplot
scales      xyplotxyzscales
      arrows=FALSEdraw=FALSEdistancescalesscales$z
      scalesxyplot
      scalesscales.3dpanel.cloud
axis.default  cloudwireframeaxis.defaultcloud
zlab      xlabylabzlabxlabylabrot
zlim      xlimylim
drape      TRUElevelplotshade = TRUEpanel.3dwire
atcol.regionsalpha.regions
      levelplotdrape=TRUEatcol.regionsalpha.regionscol.regions
      alpha.regions"regions"
cuts      atdrape=TRUE
pretty
colorkey      levelplot
...      distanceperspectivescreenR.matshadewireframepanel.cloud
      zoom

```

```

aspectscreen1/distanceperspective=FALSEdistance
cloudwireframewireframegroupsgroupsccloudpanel.superposepanel.3dscatter
wireframepanel.3dwirecloudwireframe
wireframewireframexz~x*xyz
drapez(x,y,z)
groupssubscriptssubset
panel.cloudscpospanel.cloud
xyplot

```

```

"trellis"updateprint

```

```

wireframegroupsNAz

```

```

<Deepayan.Sarkar@R-project.org>

```

```

http://lmdvr.r-forge.r-project.org/

```

```

Latticexyplotlevelplotpanel.cloud
panel.identify.cloud

```

```

## volcano ## 87 x 61 matrix
wireframe(volcano, shade = TRUE,
           aspect = c(61/87, 0.4),
           light.source = c(10,0,10))

g <- expand.grid(x = 1:10, y = 5:15, gr = 1:2)
g$z <- log((g$x^g$gr + g$y^2) * g$gr)
wireframe(z ~ x * y, data = g, groups = gr,
           scales = list(arrows = FALSE),
           drape = TRUE, colorkey = TRUE,
           screen = list(z = 30, x = -60))

cloud(Sepal.Length ~ Petal.Length * Petal.Width | Species, data = iris,
      screen = list(x = -90, y = 70), distance = .4, zoom = .6)

## cloud.table

cloud(prop.table(Titanic, margin = 1:3),
      type = c("p", "h"), strip = strip.custom(strip.names = TRUE),
      scales = list(arrows = FALSE, distance = 2), panel.aspect = 0.7,
      zlab = "Proportion")[, 1]

## transparent axes

```

```

par.set <-
  list(axis.line = list(col = "transparent"),
        clip = list(panel = "off"))
print(cloud(Sepal.Length ~ Petal.Length * Petal.Width,
            data = iris, cex = .8,
            groups = Species,
            main = "Stereo",
            screen = list(z = 20, x = -70, y = 3),
            par.settings = par.set,
            scales = list(col = "black")),
      split = c(1,1,2,1), more = TRUE)
print(cloud(Sepal.Length ~ Petal.Length * Petal.Width,
            data = iris, cex = .8,
            groups = Species,
            main = "Stereo",
            screen = list(z = 20, x = -70, y = 0),
            par.settings = par.set,
            scales = list(col = "black")),
      split = c(2,1,2,1))

```

B.08 splom

```

splom(x, data, ...)
parallelplot(x, data, ...)

## S3 method for class 'formula'
splom(x,
      data,
      auto.key = lattice.getOption("default.args")$auto.key,
      aspect = 1,
      between = list(x = 0.5, y = 0.5),
      panel = lattice.getOption("panel.splom"),
      prepanel,
      scales,
      strip,
      groups,
      xlab,
      xlim,
      ylab = NULL,
      ylim,
      superpanel = lattice.getOption("panel.pairs"),
      pscales = 5,
      varnames = NULL,
      drop.unused.levels,
      ...,
      lattice.options = NULL,

```

```

        default.scales,
        default.prepanel = lattice.getOption("prepanel.default.splom"),
        subset = TRUE)
## S3 method for class 'formula'
parallelplot(x,
             data,
             auto.key = lattice.getOption("default.args")$auto.key,
             aspect = "fill",
             between = list(x = 0.5, y = 0.5),
             panel = lattice.getOption("panel.parallel"),
             prepanel,
             scales,
             strip,
             groups,
             xlab = NULL,
             xlim,
             ylab = NULL,
             ylim,
             varnames = NULL,
             horizontal.axis = TRUE,
             drop.unused.levels,
             ...,
             lattice.options = NULL,
             default.scales,
             default.prepanel = lattice.getOption("prepanel.default.parallel"),
             subset = TRUE)

## S3 method for class 'data.frame'
splom(x, data = NULL, ..., groups = NULL, subset = TRUE)
## S3 method for class 'matrix'
splom(x, data = NULL, ..., groups = NULL, subset = TRUE)

## S3 method for class 'matrix'
parallelplot(x, data = NULL, ..., groups = NULL, subset = TRUE)
## S3 method for class 'data.frame'
parallelplot(x, data = NULL, ..., groups = NULL, subset = TRUE)

```

```

x
    "formula"~ x | g1 * g2 * ...xg1,g2,...g1, g2, ...
    data.frame
data    formulagroupssubset
aspect    splom
between
panel    parallelplot
         splomsuperpanelpanelsuperpanelpanel.pairspanel.pairs
superpanel
pscales    scalesxyplotsuperpanelpanel.pairs
varnames    xx

```

```

horizontal.axis
    TRUEFALSE
auto.keyprepanelscalesstripgroupsxlabxlimylabylimdrop.unused.levels
lattice.optionsdefault.scalessubset
    xyplot
default.prepanel
    xyplot
...
    xyplot

splompanelsuperpanel0.5ncol(z) + 0.5panel.pairs(i, j)zc(i, j)panel.pairs
xyplotpanel.pairssplomsplompanel.pairsscalessplom
parallelplot
xyplot

"trellis"updateprint

<Deepayan.Sarkar@R-project.org>

```

```

xyplotLatticepanel.pairspanel.parallel

```

```

super.sym <- trellis.par.get("superpose.symbol")
splom(~iris[1:4], groups = Species, data = iris,
      panel = panel.superpose,
      key = list(title = "Three Varieties of Iris",
                  columns = 3,
                  points = list(pch = super.sym$pch[1:3],
                                col = super.sym$col[1:3]),
                  text = list(c("Setosa", "Versicolor", "Virginica"))))
splom(~iris[1:3]|Species, data = iris,
      layout=c(2,2), pscales = 0,
      varnames = c("Sepal\nLength", "Sepal\nWidth", "Petal\nLength"),
      page = function(...) {
        ltext(x = seq(.6, .8, length.out = 4),
              y = seq(.9, .6, length.out = 4),
              labels = c("Three", "Varieties", "of", "Iris"),
              cex = 2)
      })
parallelplot(~iris[1:4] | Species, iris)
parallelplot(~iris[1:4], iris, groups = Species,
             horizontal.axis = FALSE, scales = list(x = list(rot = 90)))

```

```
tmdxyplotqqqmathformuladata.frame tmd tmdxyplot
```

```
tmd(object, ...)
```

```
## S3 method for class 'trellis'
```

```
tmd(object,  
     xlab = "mean",  
     ylab = "difference",  
     panel,  
     prepanel,  
     ...)
```

```
prepanel.tmd.qqmath(x,  
                    f.value = NULL,  
                    distribution = qnorm,  
                    qtype = 7,  
                    groups = NULL,  
                    subscripts, ...)
```

```
panel.tmd.qqmath(x,  
                 f.value = NULL,  
                 distribution = qnorm,  
                 qtype = 7,  
                 groups = NULL,  
                 subscripts, ...,  
                 identifier = "tmd")
```

```
panel.tmd.default(x, y, groups = NULL, ...,  
                  identifier = "tmd")
```

```
prepanel.tmd.default(x, y, ...)
```

```
object          "trellis"xyplotqqqmath
```

```
xlab
```

```
ylab
```

```
panel
```

```
prepanel
```

```
f.valuedistributionqtype
```

```
panel.qqmath
```

```
groupssubscripts
```

```
xyplot
```

```
xy
```

```
...
```

```
identifier
```

```

x=(x+y)/2y=y-xy=0
tmd"trellis"tmdprepanelpaneltmd
xyplotqqdefaultqqmathprepanelpanel
tmdupdate"trellis"tmd

```

```
"trellis"updateprint
```

```
<Deepayan.Sarkar@R-project.org>
```

```
qqqqmathxyplotLattice
```

```
tmd(qqmath(~height | voice.part, data = singer))
```

B.10 rfs

```

rfs(model, layout=c(2, 1), xlab="f-value", ylab=NULL,
     distribution = qunif,
     panel, prepanel, strip, ...)

```

```

model          fitted.valuesresidualsoneway
layout
xlab            "f.value"
distribution    qqmath
ylabpanelprepanelstrip
               xyplot
...            qqmath

```

```
"trellis"updateprint
```

```
<Deepayan.Sarkar@R-project.org>
```

```
onewayqqmathxyplotLattice
```

```
rfs(oneway(height ~ voice.part, data = singer, spread = 1), aspect = 1)
```

B.11 oneway

rfs

```
oneway(formula, data, location=mean, spread=function(x) sqrt(var(x)))
```

```
formula      y ~ xyx
```

```
data
```

```
location     median
```

```
spread       sd
```

```
location
```

```
spread
```

```
fitted.values
```

```
residuals y - fitted.values
```

```
scaled.residuals spread
```

```
<Deepayan.Sarkar@R-project.org>
```

[rfsLattice](#)

C.01 trellis.device

```
trellis.device(device = getOption("device"),
               color = !(dev.name == "postscript"),
               theme = lattice.getOption("default.theme"),
               new = TRUE,
               retain = FALSE,
               ...)
```



```

device      Devices"pdf""postscript""png""jpeg""X11""windows""quartz"
color       FALSETRUEthemetrellis.par.set
theme
            lattice.options(default.theme      =      "col.whitebg")
            getOption(lattice.theme)
            theme.Device
new         FALSE
retain     TRUE
name       .Device
...        devicefileheightwidth

```

```

trellis.device
trellis.device"trellis"lattice.optionstrellis.device

```

```

trellis.device

```

```

trellis.devicebgbgtheme

```

```

<Deepayan.Sarkar@R-project.org>

```

```

http://lmdvr.r-forge.r-project.org/

```

```

Lattice lattice

```

```

Devices device

```

```

standard.theme

```

```

standard.theme(name, color = TRUE,
               symbol = palette.colors(palette = "Okabe-Ito")[c(6, 2, 4, 7, 3, 5, 8)],
               fill    = NULL,
               region = hcl.colors(14, palette = "YlGnBu", rev = TRUE),
               reference = "gray90",
               bg = "transparent",
               fg = "black",
               ...)
canonical.theme(...)
custom_theme(symbol, fill, region,
             reference = "gray90", bg = "transparent", fg = "black",
             strip.bg = rep("gray95", 7), strip.fg = rep("gray70", 7),
             ...)
classic.theme(name, color)
col.whitebg()

```

```

name          .Deviceclassic.themestandard.theme
color
symbol
fill          NULLstandard.theme
region        levelplot
reference
fg
bg
strip.bg
strip.fg
...           standard.themecustom_themesimpleTheme

```

```

trellis.par.setparlattice.optionsparpar
trellis.devicetrellis.par.set
classic.themecol.whitebg
classic.theme
standard.themeclassic.themethemelattice.options(default.theme
classic.theme("pdf"))themetrellis.device
custom_themestandard.themecanonical.themestandard.theme

```

=

```

trellis.devicethemetrellis.par.set
col.whitebgthemetrellis.devicetrellis.par.setcol.whitebg

```

<Deepayan.Sarkar@R-project.org>

<http://lmdvr.r-forge.r-project.org/>

Lattice `lattice`

Devices `device`

`trellis.par.get` `trellis.par.set` `par.settings` `xyplot` "trellis"

C.02b `trellis.par.get`

```
trellis.par.set(name, value, ..., theme, warn = TRUE, strict = FALSE)
trellis.par.get(name = NULL)
show.settings(x = NULL)
```

name	<code>trellis.par.get()</code> name
value	value
theme	<code>trellis.par.get()</code> <code>trellis.par.set</code> namevalue <code>trellis.device</code> theme
...	name = value theme = list(...)
warn	<code>trellis.par.get</code>
strict	valuevalueNULLstrict = TRUE strict1
x	themetrellis.par.get

```
lattice.theme.Device trellis.device
x11 windows postscript postscript pdf
trellis.device trellis.device
trellis.par.*
trellis.par.getname trellis.par.getnamevalue
trellis.par.get trellis.par.set
add.line <- trellis.par.get("add.line")
add.line$col <- "red"
trellis.par.set("add.line", add.line)
```

```
trellis.par.set(list(add.line = list(col = "red")))
```

```
trellis.par.set(add.line = list(col = "red"))
trellis.settingsprint(trellis.par.get())
show.settings
```

```
trellis.par.getname
```

```
grid.pars  
fontsize textpoints  
clip "on""off"panelstrip  
axis.components lefttoprightbottomtckpad1pad2  
layout.heights  
layout.widths
```

```
trellis.par.gettrellis.par.setparpargrid.parsgpar
```

```
<Deepayan.Sarkar@R-project.org>
```

```
trellis.deviceLatticegpar
```

```
show.settings()
```

```
tp <- trellis.par.get()
```

```
unusual <- c("grid.pars", "fontsize", "clip", "axis.components",  
            "layout.heights", "layout.widths")
```

```
for (u in unusual) tp[[u]] <- NULL  
names.tp <- lapply(tp, names)  
unames <- sort(unique(unlist(names.tp)))  
ans <- matrix(0, nrow = length(names.tp), ncol = length(unames))  
rownames(ans) <- names(names.tp)  
colnames(ans) <- unames  
for (i in seq_along(names.tp))  
  ans[i, ] <- as.numeric(unames %in% names.tp[[i]])  
ans <- ans[, order(-colSums(ans))]  
ans <- ans[order(rowSums(ans)), ]  
ans[ans == 0] <- NA
```

```
levelplot(t(ans), colorkey = FALSE,  
          scales = list(x = list(rot = 90)),  
          panel = function(x, y, z, ...) {  
            panel.abline(v = unique(as.numeric(x)),  
                          h = unique(as.numeric(y)),  
                          col = "darkgrey")  
            panel.xyplot(x, y, pch = 16 * z, ...)  
          },  
          xlab = "Graphical parameters",  
          ylab = "Setting names")
```

C.03 simpleTheme

par.settings

```
simpleTheme(col, alpha,  
            cex, pch, lty, lwd, font, fill, border,  
            col.points, col.line,  
            alpha.points, alpha.line)
```

```
colcol.pointscol.line  
      col"plot.symbol""plot.line""plot.polygon""superpose.symbol"  
      "superpose.line""superpose.polygon"col.pointscol"plot.symbol"  
      "superpose.symbol"col.linecol"plot.line""superpose.line"  
      "superpose"  
alphaalpha.pointsalphaline  
      col  
cexpchfont      plot.symbolsuperpose.symbol  
ltylwd          plot.linesuperpose.line  
fill            plot.symbolplot.polygonsuperpose.symbolsuperpose.polygon  
border          plot.polygonsuperpose.polygon
```

```
trellis.deviceauto.keypar.settingssimpleThemename=value
```

```
themetrellis.devicetrellis.par.setpar.settingsxyplot
```

<Deepayan.Sarkar@R-project.org>

```
trellis.devicexyplotLattice
```

```
str(simpleTheme(pch = 16))
```

```
dotplot(variety ~ yield | site, data = barley, groups = year,  
         auto.key = list(space = "right"),  
         par.settings = simpleTheme(pch = 16),  
         xlab = "Barley Yield (bushels/acre) ",  
         aspect=0.5, layout = c(1,6))
```

```
lattice.options(...)
lattice.getOption(name)
```

```
name
...           name = value
```

```
optionsgetOption
```

```
panel.error NULLtryCatchpanel.errorNULLtryCatch
save.object "trellis"TRUE
layout.widthslayout.heights "trellis"unitxunitsdata
    trellis.par.set
drop.unused.levels conddataxyplot
legend.bbox "full""panel"xyspace="inside"keyxyplot
default.args as.tableauto.keyaspectbetweengridskipstripxscale.components
    yscale.componentsaxis
highlight.gpar gpartrellis.focus
banking banking
axis.padding "numeric""factor"
skip.boundary.labels
interaction.sep xyplot
optimize.grid FALSE
axis.units

panel.xyplotprepanel.default.xyplotxyplot
```

```
lattice.getOptionlattice.optionslattice.options
```

```
<Deepayan.Sarkar@R-project.org>
```

```
optionstrellis.devicetrellis.par.getLattice
```

```

names(lattice.options())
str(lattice.getOption("layout.widths"), max.level = 2)

## Not run:
## change default settings for subsequent plots
lattice.options(default.args = list(as.table = TRUE,
                                   grid = TRUE,
                                   auto.key = TRUE))

## End(Not run)

```

C.05 print.trellis

```

printplot"trellis"printplotsummarydimdimnamespanel.error

## S3 method for class 'trellis'
plot(x, position, split,
     more = FALSE, newpage = TRUE,
     packet.panel = packet.panel.default,
     draw.in = NULL,
     panel.height = lattice.getOption("layout.heights")$panel,
     panel.width = lattice.getOption("layout.widths")$panel,
     save.object = lattice.getOption("save.object"),
     panel.error = lattice.getOption("panel.error"),
     prefix,
     ...)
## S3 method for class 'trellis'
print(x, ...)

## S3 method for class 'trellis'
summary(object, ...)

## S3 method for class 'trellis'
dim(x)
## S3 method for class 'trellis'
dimnames(x)

panel.error(e)

xobject      "trellis"
position
split
more

```

```

newpage
packet.panel    packet.panel.defaultpacket.panel
draw.in        namedownViewportnewpage
panel.widthpanel.height
                xunitsunit()dataunit
                aspect
save.object     updatetrellis.focus
panel.error     tryCatchpanel.errorpanel.error = stop
                tryCatchtryCatchpanel.error
prefix         "trellis""plot_01""plot_02""::"
e              tryCatch
...            printplotprint

```

```

"trellis"xyplotbwplotsplitposition
newpage = FALSEdraw.in
x
trellis.focus

```

```

positionsplit

```

```

<Deepayan.Sarkar@R-project.org>

```

[Latticeunitupdate.trellistrellis.focuspacket.panel.default](#)

```

p11 <- histogram( ~ height | voice.part, data = singer, xlab="Height")
p12 <- densityplot( ~ height | voice.part, data = singer, xlab = "Height")
p2 <- histogram( ~ height, data = singer, xlab = "Height")

```

```

## simple positioning by split
print(p11, split=c(1,1,1,2), more=TRUE)
print(p2, split=c(1,2,1,2))

```

```

## Combining split and position:
print(p11, position = c(0,0,.75,.75), split=c(1,1,1,2), more=TRUE)
print(p12, position = c(0,0,.75,.75), split=c(1,2,1,2), more=TRUE)
print(p2, position = c(.5,.75,1,1), more=FALSE)

```

```

## Using seekViewport

```

```

## repeat same plot, with different polynomial fits in each panel
xyplot(Armed.Forces ~ Year, longley, index.cond = list(rep(1, 6)),
       layout = c(3, 2),
       panel = function(x, y, ...)

```



```

    {
      panel.xyplot(x, y, ...)
      fm <- lm(y ~ poly(x, panel.number()))
      llines(x, predict(fm))
    })

## Not run:
grid::seekViewport(trellis.vpname("panel", 1, 1))
cat("Click somewhere inside the first panel:\n")
ltext(grid::grid.locator(), lab = "linear")

## End(Not run)

grid::seekViewport(trellis.vpname("panel", 1, 1))
grid::grid.text("linear")

grid::seekViewport(trellis.vpname("panel", 2, 1))
grid::grid.text("quadratic")

grid::seekViewport(trellis.vpname("panel", 3, 1))
grid::grid.text("cubic")

grid::seekViewport(trellis.vpname("panel", 1, 2))
grid::grid.text("degree 4")

grid::seekViewport(trellis.vpname("panel", 2, 2))
grid::grid.text("degree 5")

grid::seekViewport(trellis.vpname("panel", 3, 2))
grid::grid.text("degree 6")

```

C.06 update.trellis

"trellis"

```

## S3 method for class 'trellis'
update(object,
  panel,
  aspect,
  as.table,
  between,
  key,
  auto.key,
  legend,
  layout,
  main,
  page,
  par.strip.text,
  prepanel,
  scales,
  skip,

```

```

strip,
strip.left,
sub,
xlab,
ylab,
xlab.top,
ylab.right,
xlim,
ylim,
xscale.components,
yscale.components,
axis,
par.settings,
plot.args,
lattice.options,
index.cond,
perm.cond,
...)

## S3 method for class 'trellis'
t(x)

## S3 method for class 'trellis'
x[i, j, ..., drop = FALSE]

trellis.last.object(..., prefix)

objectx      "trellis"
ij
drop         FALSE
panelaspects.tablebetweenkeyauto.keylegendlayoutmainpagepar.strip.text
prepanelscaleskipstripstrip.leftsubxlabylabxlab.topylab.rightxlimylim
xscale.componentsyscale.componentsaxispar.settingsplot.argslattice.options
index.condperm.cond...
              object
prefix       "trellis""trellis"trellis.focus

xyplot"trellis"printupdate
updateformula, data, groups, subscriptssubset
updatexyplotpanelcloudwireframe
"["index.condtperm.cond
"trellis"trellis.last.objectprefixtrellis.last.object

trellisprint.trellistrellis.last.objectNULL

```

<Deepayan.Sarkar@R-project.org>

[trellis.objectLatticexyplot](#)

```
spots <- by(sunspots, gl(235, 12, labels = 1749:1983), mean)
old.options <- lattice.options(save.object = TRUE)
xyplot(spots ~ 1749:1983, xlab = "", type = "l",
       scales = list(x = list(alternating = 2)),
       main = "Average Yearly Sunspots")
update(trellis.last.object(), aspect = "xy")
trellis.last.object(xlab = "Year")
lattice.options(old.options)
```

C.07 shingles

```
shingle(x, intervals=sort(unique(x)))
equal.count(x, ...)
as.shingle(x)
is.shingle(x)

## S3 method for class 'shingle'
plot(x, panel, xlab, ylab, ...)

## S3 method for class 'shingle'
print(x, showValues = TRUE, ...)

## S3 method for class 'shingleLevel'
as.character(x, ...)

## S3 method for class 'shingleLevel'
print(x, ...)

## S3 method for class 'shingle'
summary(object, showValues = FALSE, ...)

## S3 method for class 'shingle'
x[subset, drop = FALSE]
as.factorOrShingle(x, subset, drop)
```

```
x          plot.shinglex[]print.shingleLevel
object
showValues
intervals
subset
drop
panelxlabylab  xyplot
...           equal.countco.intervalsplot
```

```
levelsnlevels
levels()as.character
equal.countxco.intervalsco.intervals
shingleintervalsintervals
as.shingleshingle(x)x
is.shinglex
plot.shingleprint.shinglesummary.shingle
```

```
x$intervalslevels.shingle(x)is.shingle"trellis"plotprint.trellis"shingle"
```

```
<Deepayan.Sarkar@R-project.org>
```

```
xyplotco.intervalsLattice
```

```
z <- equal.count(rnorm(50))
plot(z)
print(z)
print(levels(z))
```

D draw.colorkey

```
levelplot
```

```
draw.colorkey(key, draw = FALSE, vp = NULL)
```

key [levelplotcolorkey](#)

draw

vp

"grob"

<Deepayan.Sarkar@R-project.org>

[xyplotlevelplot](#)

D draw.key

draw.key(key, draw=FALSE, vp=NULL, ...)

key [xyplotkey](#)

draw

vp

...

<Deepayan.Sarkar@R-project.org>

[xyplot](#)

D `level.colors`

```
level.colors(x, at, col.regions, colors = TRUE, ...)
```

```
x          factor
at          x
col.regions at
colors      FALSEatx
...
```

```
atxatNA
```

```
col.regionstopo.colorscolorRampPalettecol.regionscol.regionscol.regions
```

```
xcolorsx
```

```
<deepayan.sarkar@r-project.org>
```

```
levelplotcolorRampPalette
```

```
depth.col <-
  with(quakes,
    level.colors(depth, at = do.breaks(range(depth), 30),
      col.regions = hcl.colors))
```

```
xyplot(lat ~ long | equal.count(stations), quakes,
  strip = strip.custom(var.name = "Stations"),
  colours = depth.col,
  panel = function(x, y, colours, subscripts, ...) {
    panel.xyplot(x, y, pch = 21, col = "transparent",
      fill = colours[subscripts], ...)
  })
```

D make.groups

```
make.groups(...)
```

```
...
```

```
data
```

```
which data
```

```
rbindwhich
```

```
<Deepayan.Sarkar@R-project.org>
```

```
Lattice
```

```
sim.dat <-  
  make.groups(uniform = runif(200),  
              exponential = rexp(175),  
              lognormal = rlnorm(150),  
              normal = rnorm(125))  
qqmath( ~ data | which, sim.dat, scales = list(y = "free"))
```

D simpleKey

```
draw.key
```

```
simpleKey(text, points = TRUE,  
         rectangles = FALSE,  
         lines = FALSE,  
         col, cex, alpha, font,  
         fontface, fontfamily,  
         lineheight, ...)
```

```

text
points
rectangles
lines
colcexalphafontfontfacefontfamilylineheight
      trellis.par.get("add.text")
...      draw.key

```

```

keyxyplotsimpleKey
simpleKeykeytexttext
simpleKey"trellis"auto.keygroupstextsimpleKeylevels(groups)auto.key

```

[keyxyplot](#)

<Deepayan.Sarkar@R-project.org>

[Latticedraw.keytrellis.par.getxyplotauto.key](#)

D strip.default

```

strip.defaultstrip.defaultstrip.customstrip.default

```

```

strip.default(which.given,
              which.panel,
              var.name,
              factor.levels,
              shingle.intervals,
              strip.names = c(FALSE, TRUE),
              strip.levels = c(TRUE, FALSE),
              sep = " : ",
              style = 1,
              horizontal = TRUE,
              bg = trellis.par.get("strip.background")$col[which.given],
              fg = trellis.par.get("strip.shingle")$col[which.given],
              par.strip.text = trellis.par.get("add.text"))
strip.custom(...)

```



```

which.given
which.panel      which.packet
var.name         strip.namesstyle
factor.levels    which.givenstrip.levelsstylefactor.levelspar.strip.textxyplot
shingle.intervals
                  levels(shingle)NULL
strip.names
                  style
strip.levels
sep
style            shingle.intervals
                  style
                  stylestrip.namesstrip.levelsstyle
horizontal       horizontal=FALSEstrip.left=TRUE
par.strip.text   colcexfont
bg
fg
...              strip.default

```

```

stylexstrip = function(...) strip.default(style=2,...)strip.custom

```

```

strip.defaultstrip.customstrip.default

```

```

<Deepayan.Sarkar@R-project.org>

```

xyplotLattice

```

## Traditional use
xyplot(Petal.Length ~ Petal.Width | Species, iris,
        strip = function(..., style) strip.default(..., style = 4))

## equivalent call using strip.custom
xyplot(Petal.Length ~ Petal.Width | Species, iris,
        strip = strip.custom(style = 4))

xyplot(Petal.Length ~ Petal.Width | Species, iris,
        strip = FALSE,
        strip.left = strip.custom(style = 4, horizontal = FALSE))

```

D trellis.object

[print](#)

[xyplot](#)"class""trellis"as.tablelayoutpagepanelprepanelmainsubpar.strip.textstrip
skipxlabylabpar.settingslattice.optionsplot.args

formula

index.cond

perm.cond

aspect.fill aspect"fill"

aspect.ratio aspect.fillFALSE

call

condlevels

legend

panel.args

panel.args.common name=value

x.scales "same"

y.scales x.scales

x.between

y.between

x.limits

y.limits x.limits

packet.sizes

<Deepayan.Sarkar@R-project.org>

[Latticexyplotprint.trellis](#)


```

        prefix = lattice.getStatus("current.prefix"))
trellis.focus(name, column, row, side, clip.off,
             highlight = interactive(), ..., prefix,
             guess = TRUE, verbose = getOption("verbose"))
trellis.switchFocus(name, side, clip.off, highlight, ..., prefix)
trellis.unfocus()
trellis.panelArgs(x, packet.number)

```

```

xyz          trellis.panelArgs"trellis"
n
subscripts
labels       subscripts
distributiongroups
             panel.qqmathpanel.args
offset       offset
threshold    "points"
panel.args   xyxtrellis.focus
perspectivedistancexlimylimzlimscreenR.mataspectscales.3d
             panel.cloudpanel.identify.cloud
panel.3d.identify
             panel.identify
name
             trellis.vpnametrellis.focustrellis.grobname
             namecolumnrowtrellis.focus
             name"panel""strip""strip.left"columnrowname"legend"side
columnrow    as.table=TRUE
guess        TRUEtrellis.focus
side         name="legend"
clip.off     name"panel""strip"clip.off=FALSE
type
             type"panel""strip""strip.left"
group
             group
which.givenwhich.panel
             which.paneltype"strip""strip.left"
prefix       "trellis""trellis"prefixprint.trellis
             switchFocus
highlight     trellis.focusTRUEtrellis.switchFocus
packet.number packet.number
verbose
...          panel.identify.qqmathpanel.identifypanel.identifytrellis.focus
             trellis.switchFocallattice.optionspanel.link.splom

```

```

panel.identifyidentifytrellis.focus
panel.link.splomsplomtrellis.focuspanel.brush.splompanel.link.splom
panel.identify.qqmathqqmathpanel.identify.qqmathcloudtrellis.panelArgs
panel.identifyidentifysubscriptssubscriptssubscripts = TRUExyplotssubscripts
panel.identifypanel.argsseq_along(x)

```

```

trellis.focusnametrellis.focus
trellis.focus
"trellis""trellis"trellis.currentLayout
trellis.unfocus
trellis.switchFocusrowcolumn
trellis.panelArgs

trellis.panelArgs()

"trellis"trellis.panelArgs"trellis"packet.numberpanelxyplot
trellis.vpnameotrellis.grobname

```

```

panel.identifyidentifypos=TRUE
trellis.panelArgs
trellis.vpnameotrellis.grobname
trellis.focuscolrowNULLcolrow

```

```

trellis.focuscurrent.vpTreeseekViewportdownViewporrtrellis.vpname
trellis.grobname

```

```

<Deepayan.Sarkar@R-project.org>panel.identify.qqmath

```

```

identifyLatticeprint.trellistrellis.currentLayoutcurrent.vpTreeviewports

```

```

## Not run:
xyplot(1:10 ~ 1:10)
trellis.focus("panel", 1, 1)
panel.identify()

```

```

## End(Not run)

```

```

xyplot(Petal.Length ~ Sepal.Length | Species, iris, layout = c(2, 2))
Sys.sleep(1)

```

```

trellis.focus("panel", 1, 1)

```

```

do.call("panel.lmline", trellis.panelArgs())
Sys.sleep(0.5)
trellis.unfocus()

trellis.focus("panel", 2, 1)
do.call("panel.lmline", trellis.panelArgs())
Sys.sleep(0.5)
trellis.unfocus()

trellis.focus("panel", 1, 2)
do.call("panel.lmline", trellis.panelArgs())
Sys.sleep(0.5)
trellis.unfocus()

## choosing loess smoothing parameter

p <- xyplot(dist ~ speed, cars)

panel.loessresid <-
  function(x = panel.args$x,
           y = panel.args$y,
           span,
           panel.args = trellis.panelArgs())
{
  fm <- loess(y ~ x, span = span)
  xgrid <- do.breaks(current.panel.limits()$xlim, 50)
  ygrid <- predict(fm, newdata = data.frame(x = xgrid))
  panel.lines(xgrid, ygrid)
  pred <- predict(fm)
  ## center residuals so that they fall inside panel
  resid <- y - pred + mean(y)
  fm.resid <- loess.smooth(x, resid, span = span)
  ##panel.points(x, resid, col = 1, pch = 4)
  panel.lines(fm.resid, col = 1)
}

spans <- c(0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)
update(p, index.cond = list(rep(1, length(spans))))
panel.locs <- trellis.currentLayout()

i <- 1

for (row in 1:nrow(panel.locs))
  for (column in 1:ncol(panel.locs))
    if (panel.locs[row, column] > 0)
    {
      trellis.focus("panel", row = row, column = column,
                    highlight = FALSE)
      panel.loessresid(span = spans[i])
      grid::grid.text(paste("span = ", spans[i]),
                      x = 0.25,
                      y = 0.75,
                      default.units = "npc")
      trellis.unfocus()
      i <- i + 1
    }

```

```
}
```

F.1 panel.barchart

barchart

```
panel.barchart(x, y, box.ratio = 1, box.width,  
               horizontal = TRUE,  
               origin = NULL, reference = TRUE,  
               stack = FALSE,  
               groups = NULL,  
               col = if (is.null(groups)) plot.polygon$col  
                 else superpose.polygon$col,  
               border = if (is.null(groups)) plot.polygon$border  
                 else superpose.polygon$border,  
               lty = if (is.null(groups)) plot.polygon$lty  
                 else superpose.polygon$lty,  
               lwd = if (is.null(groups)) plot.polygon$lwd  
                 else superpose.polygon$lwd,  
               ..., identifier = "barchart")
```

x	origin
y	
box.ratio	
box.width	box.ratio
horizontal	bwplot
origin	stack = TRUE NULLorigin = 0
reference	
stack	FALSE
groups	
colborderltylwd	plot.polygonsuperpose.polygoncolborderltylwd
...	
identifier	

barchart

<Deepayan.Sarkar@R-project.org>

barchart

```
barchart(yield ~ variety | site, data = barley,
         groups = year, layout = c(1,6), origin = 0,
         ylab = "Barley Yield (bushels/acre)",
         scales = list(x = list(abbreviate = TRUE,
                                minlength = 5)))
```

F.1 panel.bwplot

bwplot

```
panel.bwplot(x, y, box.ratio = 1,
             box.width = box.ratio / (1 + box.ratio),
             horizontal = TRUE,
             pch, col, alpha, cex,
             font, fontfamily, fontface,
             fill, varwidth = FALSE,
             notch = FALSE, notch.frac = 0.5,
             ...,
             levels.fos,
             stats = boxplot.stats,
             coef = 1.5,
             do.out = TRUE,
             identifier = "bwplot")
```

xy	yx	horizontal	TRUE	FALSE		
box.ratio						
box.width	box.ratio	box.ratio				
horizontal		bwplot				
pch	col	alpha	cex	font	fontfamily	fontface
				pch=" "	boxplot	
fill						
varwidth						
notch		notch	TRUE	boxplot.stats		
notch.frac		notch	=	TRUE		
stats		boxplot.stats	boxplot.stats	coef	do.out	...
coef	do.out	stats				
levels.fos						
...						
identifier						


```
xyhorizontal=FALSEboxplot.statsbwplot  
"box.rectangle""box.umbrella""plot.symbol"
```

<Deepayan.Sarkar@R-project.org>

[bwplotboxplot.stats](#)

```
bwplot(voice.part ~ height, data = singer,  
       xlab = "Height (inches)",  
       panel = function(...) {  
         panel.grid(v = -1, h = 0)  
         panel.bwplot(...)  
       },  
       par.settings = list(plot.symbol = list(pch = 4)))  
  
bwplot(voice.part ~ height, data = singer,  
       xlab = "Height (inches)",  
       notch = TRUE, pch = "|")
```

F.1 panel.cloud

cloudwireframe

```
panel.cloud(x, y, subscripts, z,  
            groups = NULL,  
            perspective = TRUE,  
            distance = if (perspective) 0.2 else 0,  
            xlim, ylim, zlim,  
            panel.3d.cloud = "panel.3dscatter",  
            panel.3d.wireframe = "panel.3dwire",  
            screen = list(z = 40, x = -60),  
            R.mat = diag(4), aspect = c(1, 1),  
            par.box = NULL,  
            xlab, ylab, zlab,  
            xlab.default, ylab.default, zlab.default,  
            scales.3d,  
            proportion = 0.6,  
            wireframe = FALSE,  
            scpos,  
            ...,  
            at,  
            identifier = "cloud")
```

```

panel.wireframe(...)
panel.3dscatter(x, y, z, rot.mat, distance,
               groups, type = "p",
               xlim, ylim, zlim,
               xlim.scaled, ylim.scaled, zlim.scaled,
               zero.scaled,
               col, col.point, col.line,
               lty, lwd, cex, pch, fill,
               cross, ..., .scale = FALSE, subscripts,
               identifier = "3dscatter")
panel.3dwire(x, y, z, rot.mat = diag(4), distance,
            shade = FALSE,
            shade.colors.palette = trellis.par.get("shade.colors")$palette,
            light.source = c(0, 0, 1000),
            xlim, ylim, zlim,
            xlim.scaled,
            ylim.scaled,
            zlim.scaled,
            col = if (shade) "transparent" else "black",
            lty = 1, lwd = 1,
            alpha,
            col.groups = superpose.polygon$col,
            polynum = 100,
            ...,
            .scale = FALSE,
            drape = FALSE,
            at,
            col.regions = regions$col,
            alpha.regions = regions$alpha,
            identifier = "3dwire")
makeShadePalette(col.regions, ..., min = 0.05, pref = 0.75)

```

xyz	panel.cloudformulaxypanel.3dscatterpanel.3dwiresubscripts panel.3dwirexyzz panel.cloudwireframepanel.3dwirexyz
subscripts	xyzpanel.cloudsubscripts
groups	
perspective	FALSEdistance
distance	1 / distancecloud
screen	"x""y""z"
R.mat	screen
par.box	box.3d
xlimylimzlim	
panel.3d.cloudpanel.3d.wireframe	cloudwireframe xlimylimzlimxlim.scaledylim.scaledzlim.scaled
aspect	cloud
xlabylabzlab	cloud

```

xlab.default
ylab.default
zlab.default
scales.3d
proportion
scpos          panel.cloud
wireframe
drape          levelplotshade=TRUE
at             drape = TRUEwireframelevelplotat
col.regions    atdrape = TRUE
               makeShadePalettec.col.regions
alpha.regions  drape = TRUE
rot.mat        screenR.matpanel.cloud
type           "p""l""h""b""p""l""b""h"z = 0
xlim.scaledylim.scaledzlim.scaled

zero.scaled    type = "h"
cross          TRUEpch = "+"panel.3dscatterpch
               FALSE"+"

shade
shade.colors.palette

               shade.colors.palettemakeShadePalette
min            makeShadePalette
pref
light.source
polynum        polynumgrid.polygon
col.groups
col.col.pointcol.linetylwdcxpcfillalpha
               lex...

...
.scale         xyzxyz.scale=TRUEpanel.3dscatterpanel.3dwire
identifier

cloudwireframepanel.wireframepanel.cloud
panel.cloudwireframepanel.3d.wireframepanel.3d.cloud
xyzpanel.3d.cloudxyzlength(x) * length(y)
panel.3dscatterpanel.3d.cloudtypepanel.xyplot
panel.3dwirepanel.3d.wireframepolynumgrid.polygonshade    =    TRUElight.source
palette.shade
groupswireframe

```

<Deepayan.Sarkar@R-project.org>

[cloudutilities.3d](#)

```
wireframe(volcano, shade = TRUE,
          shade.colors.palette = makeShadePalette(hcl.colors(10, "Inferno"),
                                                  pref = 0.2))
wireframe(volcano, shade = TRUE,
          shade.colors.palette = makeShadePalette(hcl.colors(10, "Dark Mint"),
                                                  pref = 0.2))
wireframe(volcano, shade = TRUE,
          shade.colors.palette = makeShadePalette(hcl.colors(10, "Harmonic"),
                                                  pref = 0.2))
```

F.1 panel.densityplot

[densityplot](#)

```
panel.densityplot(x, darg, plot.points = "jitter",
                  ref = FALSE,
                  groups = NULL,
                  weights = NULL,
                  jitter.amount,
                  type, ...,
                  grid = lattice.getOption("default.args")$grid,
                  identifier = "density")

x
darg      densitybwadjustkernelwindowwidthgive.Rkernnnfromtocutna.rm
          density
plot.points  "rug"panel.rug"jitter"
ref
groups      panel.superpose
weights     ...subscriptsx
jitter.amount plot.points="jitter"amountjitter
type        type
...         panel.rugpanel.densityplot
grid        gridpanel.grid
            TRUE list(h = -1, v = -1)
            "h" list(h = -1, v = 0)
            "v" list(h = 0, v = -1)
            grid = FALSE
identifier
```

<Deepayan.Sarkar@R-project.org>

[densityplotjitter](#)

F.1 `panel.dotplot`

`dotplot`

```
panel.dotplot(x, y, horizontal = TRUE,
              pch, col, lty, lwd, col.line,
              levels.fos,
              groups = NULL,
              ...,
              grid = lattice.getOption("default.args")$grid,
              identifier = "dotplot")
```

`xy`

`horizontal` [bwplot](#)
`pch``col``lty``lwd``col.line`

`levels.fos`

`groups`

`...` `panel.xyplot``panel.dotplot`

`grid` `grid`[panel.grid](#)`FALSE``grid = TRUE``list(h = 0, v = -1)``horizontal =`
 `TRUE``list(h = -1, v = 0)``horizontal = FALSE``grid`

`identifier`

`xy`

<Deepayan.Sarkar@R-project.org>

[dotplot](#)

F.1 `panel.histogram`

`histogram`

```
panel.histogram(x,
               breaks,
               equal.widths = TRUE,
               type = "density",
               nint = round(log2(length(x)) + 1),
               alpha, col, border, lty, lwd,
               ...,
               identifier = "histogram")
```

```
x
breaks
equal.widths    breaks==NULL
type            "percent""density""count"
nint
alpha,col,border,lty,lwd
               plot.polygon
...            hist
identifier
```

<Deepayan.Sarkar@R-project.org>

[histogram](#)

F.1 `panel.levelplot`

[levelplot](#)`contourplot``levelplot`

```
panel.levelplot(x, y, z,
               subscripts,
               at = pretty(z),
               shrink,
               labels,
               label.style = c("mixed", "flat", "align"),
               contour = FALSE,
```

```

        region = TRUE,
        col = add.line$col,
        lty = add.line$lty,
        lwd = add.line$lwd,
        border = "transparent",
        border.lty = 1,
        border.lwd = 0.1,
        ...,
        region.type = c("grid", "contour"),
        col.regions = regions$col,
        alpha.regions = regions$alpha,
        identifier = "levelplot")
panel.contourplot(...)

panel.levelplot.raster(x, y, z,
                      subscripts,
                      at = pretty(z),
                      ...,
                      col.regions = regions$col,
                      alpha.regions = regions$alpha,
                      interpolate = FALSE,
                      identifier = "levelplot")

```

xyz	
subscripts	xyz
at	zlevelplot
shrink	
labels	atlabels
	labels
	col, cex, alpha
	fontfamily, fontface, font
label.style	"flat""align"
contour	
region	
colltylwd	
border	region=TRUE
border.ltyborder.lwd	
...	
region.type	"grid""contour" filled.contour "contour"xyz
col.regions	region=TRUEat level.colors
alpha.regions	
interpolate	grid.raster
identifier	

```
levelplotcontourplotpanel.contourplotpanel.levelplot
contour=TRUEcontourLines
panel.levelplot.rastergrid.rasterpanel.levelplot.rasterlevelplot
```

```
<Deepayan.Sarkar@R-project.org>
region.type = "contour"
```

```
levelplotlevel.colorscontourLinesfilled.contour
```

```
require(grid)

levelplot(rnorm(10) ~ 1:10 + sort(runif(10)), panel = panel.levelplot)

suppressWarnings(plot(levelplot(rnorm(10) ~ 1:10 + sort(runif(10))),
                             panel = panel.levelplot.raster,
                             interpolate = TRUE)))

levelplot(volcano, panel = panel.levelplot.raster)

levelplot(volcano, panel = panel.levelplot.raster,
           col.regions = hcl.colors, cuts = 30, interpolate = TRUE)
```

F.1 panel.pairs

splom

```
panel.pairs(z,
            panel = lattice.getOption("panel.splom"),
            lower.panel = panel,
            upper.panel = panel,
            diag.panel = "diag.panel.splom",
            as.matrix = FALSE,
            groups = NULL,
            panel.subscripts,
            subscripts,
            pscales = 5,
            prepanel.limits = scale_limits,
            varnames = colnames(z),
            varname.col, varname.cex, varname.font,
            varname.fontfamily, varname.fontface,
            axis.text.col, axis.text.cex, axis.text.font,
            axis.text.fontfamily, axis.text.fontface,
```



```

axis.text.lineheight,
axis.line.col, axis.line.lty, axis.line.lwd,
axis.line.alpha, axis.line.tck,
...)
diag.panel.splom(x = NULL,
varname = NULL, limits, at = NULL, labels = NULL,
draw = TRUE, tick.number = 5,
varname.col, varname.cex,
varname.lineheight, varname.font,
varname.fontfamily, varname.fontface,
axis.text.col, axis.text.alpha,
axis.text.cex, axis.text.font,
axis.text.fontfamily, axis.text.fontface,
axis.text.lineheight,
axis.line.col, axis.line.alpha,
axis.line.lty, axis.line.lwd,
axis.line.tck,
...)

z
panellower.panelupper.panel
lower.panelupper.panel
panel.pairsij
diag.panel diag.panel.splomdiag.panel=NULL
as.matrix TRUEpairsas.table
groups
panel.subscripts
subscripts
subscripts z
pscales splompscalespscalesz
at
labels
limits
pscales=0
prepanel.limits
xxlimylimxyplot
xyplotprepanellimitspscales

x
varname
limits
at
labels
draw FALSE
tick.number
varnames xx
varname.col gpar

```

```
varname.cex  
varname.lineheight
```

```
varname.fontvarname.fontfamilyvarname.fontface
```

```
axis.text.col  
axis.text.cex  
axis.text.fontaxis.text.fontfamilyaxis.text.fontface
```

```
axis.text.lineheight
```

```
axis.text.alpha
```

```
axis.line.col  
axis.line.lty  
axis.line.lwd  
axis.line.alpha
```

```
axis.line.tck
```

```
...          panellower.panelupper.paneldiag.panelpanel.pairs  
              diag.panel.splom
```

```
panel.pairs"trellis"splom
```

```
<Deepayan.Sarkar@R-project.org>
```

[splom](#)

```
Cmat <- outer(1:6,1:6,  
              function(i,j) hcl.colors(11)[i+j-1]) ## rainbow(11, start=.12, end=.5)[i+j-1])  
  
splom(~diag(6), as.matrix = TRUE,  
      panel = function(x, y, i, j, ...) {  
        panel.fill(Cmat[i,j])  
        panel.text(.5,.5, paste("(",i,"",j,")",sep=""))  
      })
```

F.1 panel.parallel

parallel

```
panel.parallel(x, y, z, subscripts,
               groups = NULL,
               col, lwd, lty, alpha,
               common.scale = FALSE,
               lower,
               upper,
               ...,
               horizontal.axis = TRUE,
               identifier = "parallel")
```

xy

z

subscripts z

groups z

collwdltyalpha superpose.linegroupsz

common.scale zFALSElowerupper

lowerupper zz

...

horizontal.axis

TRUEFALSE

identifier

[parallel](#)

<Deepayan.Sarkar@R-project.org>

[parallel](#)

F.1 panel.qqmath

qqmath

```
panel.qqmath(x, f.value = NULL,  
             distribution = qnorm,  
             qtype = 7,  
             groups = NULL, ...,  
             tails.n = 0,  
             identifier = "qqmath")
```

```
x  
f.value  
distribution  
             qqmath  
qtype             typequantile  
groups  
...               panel.xyplotgridablinepanel.xyplot  
tails.n           f.value = NULLf.value = NULLtails.nqtype  
identifier
```

distributionqqmath

<Deepayan.Sarkar@R-project.org>

qqmath

```
set.seed(0)  
xx <- rt(10000, df = 10)  
qqmath(~ xx, pch = "+", distribution = qnorm,  
       grid = TRUE, abline = c(0, 1),  
       xlab.top = c("raw", "ppoints(100)", "tails.n = 50"),  
       panel = function(..., f.value) {  
         switch(panel.number(),  
               panel.qqmath(..., f.value = NULL),  
               panel.qqmath(..., f.value = ppoints(100)),  
               panel.qqmath(..., f.value = ppoints(100), tails.n = 50))  
       }, layout = c(3, 1))[c(1,1,1)]
```

F.1 panel.stripplot

stripplotpanel.superpose

```
panel.stripplot(x, y, jitter.data = FALSE,  
               factor = 0.5, amount = NULL,  
               horizontal = TRUE, groups = NULL,  
               ...,  
               grid = lattice.getOption("default.args")$grid,  
               identifier = "stripplot")
```

xy

jitter.data [panel.xyplot](#)jitter.xjitter.yhorizontal

factoramount [jitter](#)

horizontal [bwplot](#)

groups

... [panel.xyplot](#)

grid [grid](#)[panel.grid](#)

TRUE list(h = -1, v = -1)

"h" list(h = -1, v = 0)

"v" list(h = 0, v = -1)

grid = FALSE

identifier

xyhorizontal

<Deepayan.Sarkar@R-project.org>

[stripplotjitter](#)

F.1 panel.xyplot

xyplotpanel.superposesplomqq

```
panel.xyplot(x, y, type = "p",
             groups = NULL,
             pch, col, col.line, col.symbol,
             font, fontfamily, fontface,
             lty, cex, fill, lwd,
             horizontal = FALSE, ...,
             smooth = NULL,
             grid = lattice.getOption("default.args")$grid,
             abline = NULL,
             jitter.x = FALSE, jitter.y = FALSE,
             factor = 0.5, amount = NULL,
             identifier = "xyplot")
panel.splom(..., identifier = "splom")
panel.qq(..., identifier = "qq")
```

xy

```
type          xy"p""l""h""b""o""s""S""g""r""a""smooth""spline"type
               typeplot"p""l""b""o""h"horizontal = TRUE"s""S""l""s""S""s"
               "S"horizontalplottype = "s"panel = panel.points
               "g"panel.gridgrid
               typesmoothsmoothtypesmoothtype
               "r""smooth""spline""a"yxsmooth
               example(xyplot)demo(lattice)
groups        panel.superpose
colcol.linecol.symbol
               plot.symbolplot.linetrellis.par.get
fontfontfacefontfamily
               pch
pchltycexlwdfill
               fillbgpointspch
horizontal    type"h""s""S"
...           panel.xyplot
smooth        "lm""loess""spline""average"TRUE"loess"smooth
               "lm"panel.lmline"loess"panel.loess"spline"panel.spline"average"
               panel.averagegroups
               yxyhorizontal = TRUE
grid          type="g"type="g"typegrid
               gridpanel.grid
               TRUE list(h = -1, v = -1)
```

```

      "h" list(h = -1, v = 0)
      "v" list(h = 0, v = -1)
      grid = FALSE
abline      panel.ablineablinepanel.ablineabline = c(0, 1)abline = coef(fm)
abline = list(h = 0, v = 0, col = "grey")
      panel.abline
jitter.xjitter.y

factoramount
identifier

```

```

xypanel.qqpanel.xyplot
xyplot

```

<Deepayan.Sarkar@R-project.org>

[panel.superposexyplotsplom](#)

```

types.plain <- c("p", "l", "o", "r", "g", "s", "S", "h", "a", "smooth")
types.horiz <- c("s", "S", "h", "a", "smooth")
horiz <- rep(c(FALSE, TRUE), c(length(types.plain), length(types.horiz)))

types <- c(types.plain, types.horiz)

x <- sample(seq(-10, 10, length.out = 15), 30, TRUE)
y <- x + 0.25 * (x + 1)^2 + rnorm(length(x), sd = 5)

xyplot(y ~ x | gl(1, length(types)),
      xlab = "type",
      ylab = list(c("horizontal=TRUE", "horizontal=FALSE"), y = c(1/6, 4/6)),
      as.table = TRUE, layout = c(5, 3),
      between = list(y = c(0, 1)),
      strip = function(...) {
        panel.fill(trellis.par.get("strip.background")$col[1])
        type <- types[panel.number()]
        grid::grid.text(label = sprintf('"%s"', type),
                        x = 0.5, y = 0.5)
        grid::grid.rect()
      },
      scales = list(alternating = c(0, 2), tck = c(0, 0.7), draw = FALSE),
      par.settings =
list(layout.widths = list(strip.left = c(1, 0, 0, 0, 0))),
      panel = function(...) {
        type <- types[panel.number()]
        horizontal <- horiz[panel.number()]
        panel.xyplot(...,
                      type = type,
                      horizontal = horizontal)
      })[rep(1, length(types))]

```

F.2 llines

```
lplot.xy(xy, type, pch, lty, col, cex, lwd,
         font, fontfamily, fontface,
         col.line, col.symbol, alpha, fill,
         origin = 0, ..., identifier, name.type)

larrows(...)
llines(x, ...)
lpoints(x, ...)
lpolygon(x, ...)
lpolypath(x, ...)
lrect(...)
lsegments(...)
ltext(x, ...)

## Default S3 method:
larrows(x0 = NULL, y0 = NULL, x1, y1, x2 = NULL, y2 = NULL,
        angle = 30, code = 2, length = 0.25, unit = "inches",
        ends = switch(code, "first", "last", "both"),
        type = "open",
        col = add.line$col,
        alpha = add.line$alpha,
        lty = add.line$lty,
        lwd = add.line$lwd,
        fill = NULL,
        font, fontface,
        ..., identifier, name.type)
## Default S3 method:
llines(x, y = NULL, type = "l",
       col, alpha, lty, lwd, ..., identifier, name.type)
## Default S3 method:
lpoints(x, y = NULL, type = "p", col, pch, alpha, fill,
        font, fontfamily, fontface, cex, ..., identifier, name.type)
## Default S3 method:
lpolygon(x, y = NULL,
        border = "black", col = "transparent", fill = NULL,
        font, fontface,
        ...,
        rule = c("none", "winding", "evenodd"),
        identifier, name.type)
## Default S3 method:
lpolypath(x, y = NULL,
         border = "black", col = "transparent", fill = NULL,
         font, fontface,
         ...,
```



```

        rule = c("winding", "evenodd"),
        identifier, name.type)
## Default S3 method:
ltext(x, y = NULL, labels = seq_along(x),
      col, alpha, cex, srt = 0,
      lineheight, font, fontfamily, fontface,
      adj = c(0.5, 0.5), pos = NULL, offset = 0.5, ..., identifier, name.type)
## Default S3 method:
lrect(xleft, ybottom, xright, ytop,
      x = (xleft + xright) / 2,
      y = (ybottom + ytop) / 2,
      width = xright - xleft,
      height = ytop - ybottom,
      col = "transparent",
      border = "black",
      lty = 1, lwd = 1, alpha = 1,
      just = "center",
      hjust = NULL, vjust = NULL,
      font, fontface,
      ..., identifier, name.type)
## Default S3 method:
lsegments(x0, y0, x1, y1, x2, y2,
          col, alpha, lty, lwd,
          font, fontface, ..., identifier, name.type)

```

```

panel.arrows(...)
panel.lines(...)
panel.points(...)
panel.polygon(...)
panel.rect(...)
panel.segments(...)
panel.text(...)

```

```

xyx0y0x1y1x2y2xy
          x2y2
lengthunit      lengthunitdataunitarrows
anglecodetypelabelssrtadjposoffset
          larrowspanel.larrowstype"open""closed"
ends            codecode
colalphatlwdfillpchcexlineheightfontfontfamilyfontfacecol.linecol.symbol
border
          fillpch21:25bgpointsalpha
          fillfontfontfacelrectlarrowslpolygonlsegmentsgpar
origin          type="h"type="H"
xleftybottomxrightytop
          rect
widthheightjusthjustvjust
          grid.rect
...

```

```
rule          NAgrid.path
              "none"NA"winding""evenodd"grid.path

identifier

name.type     "panel""strip""strip.left"""
```

```
panel.*1*
```

```
type="H"type="h"
```

```
<Deepayan.Sarkar@R-project.org>
```

```
pointslinesrecttextsegmentsarrowsLattice
```

```
SD <- 0.1
t <- seq(0, 2*pi, length.out = 50) + rnorm(50, sd = SD)
d <- list(x = c(cos(t), NA, rev(0.5 * cos(t))) + rnorm(101, sd = SD),
         y = c(sin(t), NA, rev(0.5 * sin(t))) + rnorm(101, sd = SD))

## rectangles
xyplot(y ~ x, d, panel = panel.rect, col = 4, alpha = 0.5, width = 0.1, height = 0.1)

## points and lines
xyplot(y ~ x, d, panel = panel.lines, col = 4, alpha = 0.5,
       type = "o", pch = 16)

## polygons and paths (with holes)
xyplot(y ~ x, d, panel = panel.polygon, col = 4, alpha = 0.5, rule = "evenodd")

## Example adapted from https://journal.r-project.org/articles/RJ-2012-017/
x <- c(.1, .5, .9, NA, .4, .5, .6, NA, .4, .6, .5)
y <- c(.1, .8, .1, NA, .5, .4, .5, NA, .3, .3, .2)
d <- data.frame(x = x, y = y)
xyplot(y ~ x, data = d, panel = panel.polygon, rule = "none", col = "grey")
xyplot(y ~ x, data = d, panel = panel.polypath, rule = "winding", col = "grey")
xyplot(y ~ x, data = d, panel = panel.polypath, rule = "evenodd", col = "grey")
```

F.2 panel.functions

```

panel.abline(a = NULL, b = 0,
             h = NULL, v = NULL,
             reg = NULL, coef = NULL,
             col, col.line, lty, lwd, alpha, type,
             ...,
             reference = FALSE,
             identifier = "abline")
panel.refline(...)

panel.curve(expr, from, to, n = 101,
            curve.type = "l",
            col, lty, lwd, type,
            ...,
            identifier = "curve")
panel.rug(x = NULL, y = NULL,
          regular = TRUE,
          start = if (regular) 0 else 0.97,
          end = if (regular) 0.03 else 1,
          x.units = rep("npc", 2),
          y.units = rep("npc", 2),
          col, col.line, lty, lwd, alpha,
          ...,
          identifier = "rug")
panel.average(x, y, fun = mean, horizontal = TRUE,
              lwd, lty, col, col.line, type,
              ...,
              identifier = "linejoin")
panel.linejoin(x, y, fun = mean, horizontal = TRUE,
               lwd, lty, col, col.line, type,
               ...,
               identifier = "linejoin")

panel.fill(col, border, ..., identifier = "fill")
panel.grid(h=3, v=3, col, col.line, lty, lwd, x, y, ..., identifier = "grid")
panel.lmline(x, y, ..., identifier = "lmline")
panel.mathdensity(dmath = dnorm, args = list(mean=0, sd=1),
                  n = 50, col, col.line, lwd, lty, type,
                  ..., identifier = "mathdensity")

```

xy	panel.grid	pretty
ab	panel.abline	abcoef reg
coef		
reg		coef
hv	panel.abline	
	panel.grid	h=-1 v=-1 pretty hv=1 h=1 vn=1 pretty
	xy	panel.grid
		pretty
reference	panel.abline	panel.refline
	panel.abline	reference = TRUE
expr	x	

```

n
fromto
curve.type      "p"llines
regular
startend        regular
x.unitsy.units  startendx.unitsy.unitsstartend
colcol.lineltylwdalphaborder

```

```

type            type
fun             xyhorizontalFALSEyx
horizontal      FALSExyxbwplot
dmath           xdnorm
args            dmath
...             colcol.linecol.symbol
identifier

```

```

panel.abliney = a + b * xpanel.refline
panel.grid
panel.curvecurveadd = TRUE
panel.averagexyhorizontalfunpanel.xyplotpanel.superpose
panel.linejoinpanel.average
panel.mathdensityhistogramdensityplotqqmath
panel.rugrug
panel.lmline(x, y)panel.abline(lm(y ~ x))

```

<Deepayan.Sarkar@R-project.org>

[panel.axispanel.identifyidentifytrellis.par.set](#)

Interaction Plot

```

bwplot(yield ~ site, barley, groups = year,
       panel = function(x, y, groups, subscripts, ...) {
         panel.grid(h = -1, v = 0)
         panel.stripplot(x, y, ..., jitter.data = TRUE, grid = FALSE,
                        groups = groups, subscripts = subscripts)
         panel.superpose(x, y, ..., panel.groups = panel.average, grid = FALSE,
                        groups = groups, subscripts = subscripts)
       },
       auto.key = list(points = FALSE, lines = TRUE, columns = 2))

```

Superposing a fitted normal density on a Histogram

```

histogram( ~ height | voice.part, data = singer, layout = c(2, 4),
           type = "density", border = "transparent", col.line = "grey60",
           xlab = "Height (inches)",
           ylab = "Density Histogram\n with Normal Fit",
           panel = function(x, ...) {
             panel.histogram(x, ...)
             panel.mathdensity(dmath = dnorm,
                               args = list(mean = mean(x), sd = sd(x)), ...)
           } )

```

F.2 panel.loess

```

panel.loess(x, y, span = 2/3, degree = 1,
            family = c("symmetric", "gaussian"),
            evaluation = 50,
            lwd, lty, col, col.line, type,
            horizontal = FALSE,
            ..., identifier = "loess")

```

```

xy
lwdltycolcol.line
      col.linecol
type      type
spandegreefamilyevaluation
      loess.smoothpanel.loess
horizontal  xyTRUEyxbwplot
...      panel.lines
identifier

```

[loess.smooth](#)

<Deepayan.Sarkar@R-project.org>

[loess.smoothprepanel.loess](#)

F.2 panel.qqmathline

```
panel.qqmathline(x, y = x,  
                 distribution = qnorm,  
                 probs = c(0.25, 0.75),  
                 qtype = 7,  
                 groups = NULL,  
                 ...,  
                 identifier = "qqmathline")
```

```
x          f.valueqqmath  
y          x  
distribution  
probs  
qtype      typequantile  
groups  
...  
identifier
```

<Deepayan.Sarkar@R-project.org>

[prepanel.qqmathlineqqmathquantile](#)

F.2 panel.smoothScatter

smoothScatter

```
panel.smoothScatter(x, y = NULL,  
                   nbin = 64, cuts = 255,  
                   bandwidth,  
                   col.regions,  
                   colramp,  
                   nrpoints = 100,  
                   transformation = function(x) x^0.25,  
                   pch = ".",  
                   cex = 1, col="black",  
                   range.x,  
                   ...,  
                   raster = FALSE,  
                   subscripts,  
                   identifier = "smoothScatter")
```

```

x
y          xy
nbin
cuts
bandwidth  bkde2D
col.regions colregions
colramp    nncol.regionscol.regions"white"colorRampPalette
nrpoints   nrpointsnrpoints = Inf
transformation
pchcex     nrpoints
range.x    bkde2D
col        points
...        panel.levelplot
raster     TRUEpanel.levelplot.raster
subscripts
identifier

smoothScatter

```

<deepayan.sarkar@r-project.org>

```

ddf <- as.data.frame(matrix(rnorm(40000), ncol = 4) + 1.5 * rnorm(10000))
ddf[, c(2,4)] <- (-ddf[, c(2,4)])
xyplot(V1 ~ V2 + V3, ddf, outer = TRUE,
       panel = panel.smoothScatter, aspect = "iso")
## argument to panel.levelplot
xyplot(V1 ~ V2, ddf, panel = panel.smoothScatter, cuts = 10,
       region.type = "contour")
splom(ddf, panel = panel.smoothScatter, nbin = 64, raster = TRUE)

```

F.2 panel.spline

```
panel.spline(x, y, npoints = 101,  
             lwd = plot.line$lwd,  
             lty = plot.line$lty,  
             col, col.line = plot.line$col,  
             type,  
             horizontal = FALSE, ...,  
             keep.data = FALSE,  
             identifier = "spline")
```

xy

npoints

lwdltycolcol.line

col.linecol

type type

horizontal xyTRUEyxbwplot

keep.data [smooth.spline](#)[FALSE](#)[TRUE](#)[panel.spline](#)[smooth.spline](#)

... [smooth.spline](#)[panel.lines](#)

identifier

[smooth.spline](#)

<Deepayan.Sarkar@R-project.org>

[smooth.spline](#)[prepanel.spline](#)

F.2 panel.superpose

xy


```

panel.superpose(x, y = NULL, subscripts, groups,
                panel.groups = "panel.xyplot",
                ...,
                col, col.line, col.symbol,
                pch, cex, fill, font,
                fontface, fontfamily,
                lty, lwd, alpha,
                type = "p",
                grid = lattice.getOption("default.args")$grid,
                distribute.type = FALSE)
panel.superpose.2(..., distribute.type = TRUE)

panel.superpose.plain(...,
                      col, col.line, col.symbol,
                      pch, cex, fill, font,
                      fontface, fontfamily,
                      lty, lwd, alpha)

xy
panel.groups      panel.xyplot
                  panel.groupsgroup.numbergroup.valuegroupspanel.groups
subscripts        xyxyplot
groups            groups"superpose.symbol""superpose.line"trellis.par.get
type              panel.groupspanel.groupspanel.xyplot
                  panel.superposepanel.superpose.2distribute.type
                  distribute.type = FALSEpanel.xyplottypedistribute.type = TRUE
                  typegroupstypetype
                  distribute.type = FALSE"g"typetypepanel.groups

grid              panel.xyplot
col
col.line
col.symbol
pch
cex
fill
fontfontfacefontfamily

lty
lwd
alpha
...              panel.superposepanel.superpose.2panel.groupspanel.superpose
distribute.type
                  type

```

```

panel.superposexygroups[subscripts]col.symbolpchpanel.groupsigroups
panel.groupspanel.groupscolcol.linecol.symbol"superpose.line""superpose.symbol"
colpanel.groups"black""superpose.line""superpose.symbol"alpha"superpose.line"
panel.superposepanel.superpose.2typepanel.superposepanel.superpose.2
panel.superpose.plainpanel.superposexyplot.ts

```

<Deepayan.Sarkar@R-project.org>panel.superpose.2

```

panel.groupspanel.xyplotpanel.dotplotpanel.average
Latticexyplotgroups

```

F.2 panel.violin

bwplot

```

panel.violin(x, y, box.ratio = 1, box.width,
             horizontal = TRUE,
             alpha, border, lty, lwd, col,
             varwidth = FALSE,
             bw, adjust, kernel, window,
             width, n = 50, from, to, cut,
             na.rm, ...,
             identifier = "violin")

```

xy	yx	horizontal	TRUE	FALSE
box.ratio				
box.width	box.ratio	box.ratio		
horizontal	xy	bwplot		
alpha	border	lty	lwd	col
				"plot.polygon"
varwidth	FALSE	box.ratio	TRUE	
bw	adjust	kernel	window	width
		n	from	to
		cut	na.rm	
		density		
...		density		
identifier				

xybwplot

<Deepayan.Sarkar@R-project.org>

[bwplotdensity](#)

```
bwplot(voice.part ~ height, singer,
       panel = function(..., box.ratio) {
         panel.violin(..., col = "transparent",
                     varwidth = FALSE, box.ratio = box.ratio)
         panel.bwplot(..., fill = NULL, box.ratio = .1)
       } )
```

F.3 prepanel.default

```
prepanel.default.bwplot(x, y, horizontal, nlevels, origin, stack, ...)
prepanel.default.histogram(x, breaks, equal.widths, type, nint, ...)
prepanel.default.qq(x, y, ...)
prepanel.default.xyplot(x, y, type, subscripts, groups, ...)
prepanel.default.cloud(perspective, distance,
                       xlim, ylim, zlim,
                       screen = list(z = 40, x = -60),
                       R.mat = diag(4),
                       aspect = c(1, 1), panel.aspect = 1,
                       ..., zoom = 0.8)
prepanel.default.levelplot(x, y, subscripts, ...)
prepanel.default.qqmath(x, f.value, distribution, qtype,
                       groups, subscripts, ..., tails.n = 0)
prepanel.default.densityplot(x, darg, groups, weights, subscripts, ...)
prepanel.default.parallel(x, y, z, ..., horizontal.axis)
prepanel.default.splom(z, ...)
```

```
xy
horizontal
horizontal.axis
TRUEFALSE

nlevels
originstack    type="h"
breaksequal.widthstypenint
typeprepanel.default.xyplotpanel.xyplot
```

```

groupssubscripts
      xyplot
weights      subscriptsx
perspectivedistancexlimylimzlimscreenR.mataspectpanel.aspectzoom
      panel.cloud
f.valuedistributiontails.n
      panel.qqmath
darg      density
z      panel.parallelpanel.pairs
qtype      quantile
...

xlimylimdxdyxatyatxyplot

```

<Deepayan.Sarkar@R-project.org>

[xyplotbankingLattice](#)

F.3 prepanel.functions

```

prepanel.lmline(x, y, ...)
prepanel.qqmathline(x, y = x, distribution = qnorm,
                    probs = c(0.25, 0.75), qtype = 7,
                    groups, subscripts,
                    ...)
prepanel.loess(x, y, span, degree, family, evaluation,
               horizontal = FALSE, ...)
prepanel.spline(x, y, npoints = 101,
                horizontal = FALSE, ...,
                keep.data = FALSE)

xy
distribution  qqmath
qtype      quantile
probs      aspect="xy"
spandegreefamilyevaluation
      loess

```

horizontalnpoints

```
keep.data      smooth.spline
groupssubscripts
              xyplot
...           smooth.spline
```

prepanel.lmlinedxdyprepanel.qqmathlineprobsprepanel.loessprepanel.splinedxdy

xlimylimxdyxyplotprepanel.default.xyplot

<Deepayan.Sarkar@R-project.org>

xyplotbankingpanel.loesspanel.spline

G axis.default

xscale.componentsyscale.componentsaxis

```
xscale.components.default(lim,
                           packet.number = 0,
                           packet.list = NULL,
                           top = TRUE,
                           ...)
yscale.components.default(lim,
                           packet.number = 0,
                           packet.list = NULL,
                           right = TRUE,
                           ...)
axis.default(side = c("top", "bottom", "left", "right"),
             scales, components, as.table,
             labels = c("default", "yes", "no"),
             ticks = c("default", "yes", "no"),
             ..., prefix)
```

```

lim
packet.number  print.trellis0
packet.list
topright      topright
side
scales        scales
components    xscale.components.defaultyscale.components.default
as.table      as.table
labels        scales
ticks         scales
...
prefix        print.trellis

```

```

xscale.components.defaultyscale.components.defaultcomponentsaxis.default
xscale.components.default

```

```

num.limit
bottom tickslabelsticksattcktkatlabelsatlabelscheck.overlapatlabels
      check.overlap
top TRUEtopbottomFALSEtopbottom
yscale.components.defaultleftrightbottomtop

```

```

<Deepayan.Sarkar@R-project.org>

```

[Latticexyplotprint.trellis](#)

```

str(xscale.components.default(c(0, 1)))

set.seed(36872)
rln <- rlnorm(100)

densityplot(rln,
  scales = list(x = list(log = 2), alternating = 3),
  xlab = "Simulated lognormal variates",
  xscale.components = function(...) {
    ans <- xscale.components.default(...)
    ans$top <- ans$bottom
    ans$bottom$labels$labels <- parse(text = ans$bottom$labels$labels)
    ans$top$labels$labels <-

```

```

        if (require(MASS))
            fractions(2^(ans$top$labels$at))
        else
            2^(ans$top$labels$at)
    ans
  })

## Direct use of axis to show two temperature scales (Celcius and
## Fahrenheit). This does not work for multi-row plots, and doesn't
## do automatic allocation of space

F2C <- function(f) 5 * (f - 32) / 9
C2F <- function(c) 32 + 9 * c / 5

axis.CF <-
  function(side, ...)
  {
    ylim <- current.panel.limits()$ylim
    switch(side,
      left = {
        prettyF <- pretty(ylim)
        labF <- parse(text = sprintf("%s ~ degree * F", prettyF))
        panel.axis(side = side, outside = TRUE,
                    at = prettyF, labels = labF)
      },
      right = {
        prettyC <- pretty(F2C(ylim))
        labC <- parse(text = sprintf("%s ~ degree * C", prettyC))
        panel.axis(side = side, outside = TRUE,
                    at = C2F(prettyC), labels = labC)
      },
      axis.default(side = side, ...))
  }

xyplot(nhtemp ~ time(nhtemp), aspect = "xy", type = "o",
       scales = list(y = list(alternating = 3)),
       axis = axis.CF, xlab = "Year", ylab = "Temperature",
       main = "Yearly temperature in New Haven, CT")

## version using yscale.components

yscale.components.CF <-
  function(...)
  {
    ans <- yscale.components.default(...)
    ans$right <- ans$left
    ans$left$labels$labels <-
      parse(text = sprintf("%s ~ degree * F", ans$left$labels$at))
    prettyC <- pretty(F2C(ans$num.limit))
    ans$right$ticks$at <- C2F(prettyC)
    ans$right$labels$at <- C2F(prettyC)
    ans$right$labels$labels <-
      parse(text = sprintf("%s ~ degree * C", prettyC))
    ans
  }

```

```
xyplot(nhtemp ~ time(nhtemp), aspect = "xy", type = "o",
       scales = list(y = list(alternating = 3)),
       yscale.components = yscale.components.CF,
       xlab = "Year", ylab = "Temperature",
       main = "Yearly temperature in New Haven, CT")
```

G banking

```
banking(dx, dy)
```

```
dxdy
```

```
bankingaspect = "xy"xyplotdxdy
Lattice::lattice.options("banking")
```

```
<Deepayan.Sarkar@R-project.org>
```

Lattice::xyplot

```
## with and without banking
```

```
plot <- xyplot(sunspot.year ~ 1700:1988, xlab = "", type = "l",
              scales = list(x = list(alternating = 2)),
              main = "Yearly Sunspots")
print(plot, position = c(0, .3, 1, .9), more = TRUE)
print(update(plot, aspect = "xy", main = "", xlab = "Year"),
      position = c(0, 0, 1, .3))
```

```
## cut-and-stack plot (see also xyplot.ts)
```

```
xyplot(sunspot.year ~ time(sunspot.year) | equal.count(time(sunspot.year)),
       xlab = "", type = "l", aspect = "xy", strip = FALSE,
       scales = list(x = list(alternating = 2, relation = "sliced")),
       as.table = TRUE, main = "Yearly Sunspots")
```

G latticeParseFormula

xyplot

```
latticeParseFormula(model, data, dimension = 2,  
                     subset = TRUE, groups = NULL,  
                     multiple, outer,  
                     subscripts,  
                     drop)
```

```
model          dimensiony ~ x| g1 * ... *gnz ~ x * y | g1 * ...* gnyg1, ...,gn  
data  
dimension  
subset  
groups  
multipleouter  xyplot  
subscripts  
drop           drop.unused.levelsxyplot
```

```
left, right, left.name, right.name, conditionleft, right.x, right.y, left.name,  
right.x.name, right.y.name, condition
```

```
<Deepayan.Sarkar@R-project.org>
```

[xyplotLattice](#)

G packet.panel.default

"trellis"

```
packet.panel.default(layout, condlevels, page, row, column,  
                      skip, all.pages.skip = TRUE)
```

```
layout          layout
condlevels
pagerowcolumn
skip            skip
all.pages.skip skipFALSEskip
```

```
picondlevels[[i]][p[i]]
```

<Deepayan.Sarkar@R-project.org>

Latticexyplot

```
packet.panel.page <- function(n)
{
  ## returns a function that when used as the 'packet.panel'
  ## argument in print.trellis plots page number 'n' only
  function(layout, page, ...) {
    stopifnot(layout[3] == 1)
    packet.panel.default(layout = layout, page = n, ...)
  }
}

data(mtcars)
HP <- equal.count(mtcars$hp, 6)
p <-
  xyplot(mpg ~ disp | HP * factor(cyl),
         mtcars, layout = c(0, 6, 1))

print(p, packet.panel = packet.panel.page(1))
print(p, packet.panel = packet.panel.page(2))
```

G panel.axis

```
panel.axiscurrent.panel.limits
```

```
panel.axis(side = c("bottom", "left", "top", "right"),
           at,
           labels = TRUE,
           draw.labels = TRUE,
           check.overlap = FALSE,
           outside = FALSE,
```

```

        ticks = TRUE,
        half = !outside,
        which.half,
        tck = as.numeric(ticks),
        rot = if (is.logical(labels)) 0 else c(90, 0),
        text.col, text.alpha, text.cex, text.font,
        text.fontfamily, text.fontface, text.lineheight,
        line.col, line.lty, line.lwd, line.alpha)

current.panel.limits(unit = "native")

```

```

side
at
labels          atatatlabelsTRUEat
draw.labels
check.overlap  at
outside        outside=TRUE
ticks
half           splom
which.half     "lower""upper"half = TRUEsplom
tck
rot
text.col       gpar
text.alpha
text.cex
text.fonttext.fontfamilytext.fontface

text.lineheight

line.col
line.lty
line.lwd
line.alpha
unit           unit

```

```

panel.axispanel.pairssplom"trellis"

```

```

current.panel.limitsxlimylimunit"native"

```

<Deepayan.Sarkar@R-project.org>

[Latticexyplottrellis.focusunit](#)

G panel.number

panelstripaxisprepanel

current.row(prefix)
current.column(prefix)
panel.number(prefix)
packet.number(prefix)
which.packet(prefix)

trellis.currentLayout(which = c("packet", "panel"), prefix)

which
prefix "trellis""trellis"[trellis.focus](#)

trellis.currentLayoutcurrent.rowcurrent.columnpanel.numberpacket.number
which.packet

panel.numberpacket.numberpanelstrippanel.numberpacket.numberpanel.number
packet.number

<Deepayan.Sarkar@R-project.org>

[Latticexyplot](#)

G Rows

Rows(x, which)

x
which x

```
xx[[i]]x[[i]][which]
```

<Deepayan.Sarkar@R-project.org>

[xyplotLattice](#)

G utilities.3d

cloudwireframe

```
ltransform3dMatrix(screen, R.mat)
ltransform3dto3d(x, R.mat, dist)
```

x	xltransform3dto3d
screen	panel.cloud
R.mat	
dist	

```
ltransform3dMatrixltransform3dto3dcloudwireframe
screencloudwireframeR.matR.matscreen
```

<Deepayan.Sarkar@R-project.org>

[cloudpanel.cloud](#)

H barley

barley

```
"Svansota""No. 462""Manchuria""No. 475""Velvet""Peatland""Glabron""No. 457"  
  "Wisconsin No. 38""Trebi"  
19321931  
"Grand Rapids""Duluth""University Farm""Morris""Crookston""Waseca"
```

immer

```
# Graphic suggesting the Morris data switched the years 1931 and 1932  
# Figure 1.1 from Cleveland  
dotplot(variety ~ yield | site, data = barley, groups = year,  
        key = simpleKey(levels(barley$year), space = "right"),  
        xlab = "Barley Yield (bushels/acre) ",  
        aspect=0.5, layout = c(1,6), ylab=NULL)
```

H environmental

environmental

```
# Scatter plot matrix with loess lines
splom(~environmental,
      panel=function(x,y){
        panel.xyplot(x,y)
        panel.loess(x,y)
      }
)

# Conditioned plot similar to figure 5.3 from Cleveland
attach(environmental)
Temperature <- equal.count(temperature, 4, 1/2)
Wind <- equal.count(wind, 4, 1/2)
xyplot((ozone^(1/3)) ~ radiation | Temperature * Wind,
      aspect=1,
      prepanel = function(x, y)
        prepanel.loess(x, y, span = 1),
      panel = function(x, y){
        panel.grid(h = 2, v = 2)
        panel.xyplot(x, y, cex = .5)
        panel.loess(x, y, span = 1)
      },
      xlab = "Solar radiation (langleys)",
      ylab = "Ozone (cube root ppb)")
```

```
detach()

# Similar display using the coplot function
with(environmental,{
  coplot((ozone^.33) ~ radiation | temperature * wind,
    number=c(4,4),
    panel = function(x, y, ...) panel.smooth(x, y, span = .8, ...),
    xlab="Solar radiation (langleys)",
    ylab="Ozone (cube root ppb)")
})
```

H ethanol

ethanol

```
## Constructing panel functions on the fly
EE <- equal.count(ethanol$E, number=9, overlap=1/4)
xyplot(NOx ~ C | EE, data = ethanol,
  prepanel = function(x, y) prepanel.loess(x, y, span = 1),
  xlab = "Compression ratio", ylab = "NOx (micrograms/J)",
  panel = function(x, y) {
    panel.grid(h=-1, v= 2)
    panel.xyplot(x, y, grid = FALSE)
    panel.loess(x, y, span = 1)
  },
  aspect = "xy")

# Wireframe loess surface fit. See Figure 4.61 from Cleveland.
require(stats)
with(ethanol, {
```



```

eth.lo <- loess(N0x ~ C * E, span = 1/3, parametric = "C",
               drop.square = "C", family="symmetric")
eth.marginal <- list(C = seq(min(C), max(C), length.out = 25),
                    E = seq(min(E), max(E), length.out = 25))
eth.grid <- expand.grid(eth.marginal)
eth.fit <- predict(eth.lo, eth.grid)
wireframe(eth.fit ~ eth.grid$C * eth.grid$E,
          shade=TRUE,
          screen = list(z = 40, x = -60, y=0),
          distance = .1,
          xlab = "C", ylab = "E", zlab = "N0x")
})

```

H melanoma

melanoma

[melanoma](#)

<https://www.cancer.gov/>

```

# Time-series plot. Figure 3.64 from Cleveland.
xyplot(incidence ~ year,
       data = melanoma,
       aspect = "xy",
       panel = function(x, y)
         panel.xyplot(x, y, type="o", pch = 16),
       ylim = c(0, 6),
       xlab = "Year",
       ylab = "Incidence")

```

H singer

singer

Bass 2Bass 1Tenor 2Tenor 1Alto 2Alto 1Soprano 2Soprano 1

```
# Separate histogram for each voice part (Figure 1.2 from Cleveland)
histogram(~ height | voice.part,
          data = singer,
          aspect = 1,
          layout = c(2, 4),
          nint = 15,
          xlab = "Height (inches)")

# Quantile-Quantile plot (Figure 2.11 from Cleveland)
qqmath(~ height | voice.part,
       data = singer,
       aspect = 1,
       layout = c(2,4),
       prepanel = prepanel.qqmathline,
       panel = function(x, ...) {
         panel.grid()
         panel.qqmathline(x, ...)
         panel.qqmath(x, ..., grid = FALSE)
       },
       xlab = "Unit Normal Quantile",
       ylab="Height (inches)")
```

H USMortality

USMortality
USRegionalMortality

USRegionalMortality

Region

Status RuralUrban

Sex FemaleMale

Cause AlzheimersCancerCerebrovascular diseasesDiabetesFlu and pneumoniaHeart
diseaseLower respiratoryNephritisSuicideUnintentional injuries

Rate

SE

USMortalityRegion

<https://www.hhs.gov/about/agencies/iea/regional-offices/index.html>

[https://ruralhealth.und.edu/projects/health-reform-policy-research-center/
rural-urban-mortality](https://ruralhealth.und.edu/projects/health-reform-policy-research-center/rural-urban-mortality)

```
dotplot(reorder(Cause, Rate) ~ Rate | Status,  
        data = USMortality, groups = Sex, grid = FALSE,  
        par.settings = simpleTheme(pch = 16), auto.key = list(columns = 2),  
        scales = list(x = list(log = TRUE, equispaced.log = FALSE)))  
dotplot(reorder(Cause, Rate):Sex ~ Rate | Status,  
        data = USRegionalMortality, groups = Sex, auto.key = FALSE,  
        scales = list(x = list(log = TRUE, equispaced.log = FALSE)))
```

I lset

trellis.par.get/settrellis.par.set

lset(theme = col.whitebg())

theme trellis.par.get()

<Deepayan.Sarkar@R-project.org>

mgcv

anova.gam

gamgamdrop1anova.lm

```
## S3 method for class 'gam'
anova(object, ..., dispersion = NULL, test = NULL,
       freq = FALSE)
## S3 method for class 'anova.gam'
print(x, digits = max(3, getOption("digits") - 3),...)
```

object...	gamgam()
x	anova.gamanova.gam()
dispersion	
test	"Chisq""F""Cp""Chisq"NULL
freq	summary.gam
digits	

anova.glm
anova.gam
summary.gamRLRsim
print.anova.gam
freq=TRUEparaPen

anova.gamanova.glm
anova.gamsummary.gam
print.anova.gam

<simon.wood@r-project.org>

[gampredict.gamgam.checksummary.gam](#)

```
library(mgcv)
set.seed(0)
dat <- gamSim(5,n=200,scale=2)

b<-gam(y ~ x0 + s(x1) + s(x2) + s(x3),data=dat)
anova(b)
b1<-gam(y ~ x0 + s(x1) + s(x2),data=dat)
anova(b,b1,test="F")
```

bam

[family.mgcv](#)[gam](#)[bam](#)[gam](#)[bam](#)

discrete==TRUE
nthreads=cluster.Machine\$integer.max

```
bam(formula,family=gaussian(),data=list(),weights=NULL,subset=NULL,
    na.action=na.omit, offset=NULL,method="fREML",control=list(),
    select=FALSE,scale=0,gamma=1,knots=NULL,sp=NULL,min.sp=NULL,
    paraPen=NULL,chunk.size=10000,rho=0,AR.start=NULL,discrete=FALSE,
    cluster=NULL,nthreads=1,gc.level=0,use.chol=FALSE,samfrac=1,
    coef=NULL,drop.unused.levels=TRUE,G=NULL,fit=TRUE,drop.intercept=NULL,
    in.out=NULL,nei=NULL,...)
```

formula	formula.gamgam.models ste
family	glmfamily family.mgcv
data	environment(formula)gam
weights	weights <- weights/mean(weights)
subset	

```

na.action
offset      formulalmglm
method      "GCV.Cp""GACV.Cp""REML""P-REML""ML""P-ML""fREML"discrete=TRUE
            "NCV"

control      gam.control
select      gamma
scale
gamma      gammalog(n)/2n/gamma
knots      kktprs"tp"/"ts"
sp          length(sp)discrete=TRUEsp
min.sp      full.splength(min.sp)sp
paraPen      gam.models
chunk.size  4*pchunk.size < 4*pp
rho         std.rsddiscrete=TRUE
AR.start    TRUENULL
discrete    method="fREML"discreteTRUE
cluster     bamparallel
nthreads    NAmax(1,length(cluster))
gc.level
use.chol    bam
samfrac     samfrac
coef
drop.unused.levels

G           NULLbamfit=FALSEspchunk.sizegammanthreadsclusterrhogc.level
           samfracuse.cholmethodscale
fit         FALSEGbam
drop.intercept TRUE
in.out      spscale
nei         NCV
...         gam.fitmustart

```

```
discrete=FALSEbampredict.gam
```

```

gam
"tp""cr""ps"te
clustermakeClustermakeForkClusterparalleldetectCores
discrete=TRUE
discrete=TRUENCVneiigamsampleneiNCV
discrete=TRUEnthreadsparallelnthreaddsnthreads=c(2,1)
discrete=TRUEid
family.mgcvgamgam

```


"gam"gamObject

method"fREML""REML""ML"gam

"tp"

discrete=TRUEtet2

.Machine\$integer.maxgc.level=1

<simon.wood@r-project.org>

mgcv.parallelmgcv-packagemgcvObjectgam.modelssmooth.termslinear.functional.terms
stepredict.gamplot.gamsummary.gamgam.sidegam.selectiongam.controlgam.check
linear.functional.termsnegbinmagicvis.gam

library(mgcv)

See help("mgcv-parallel") for using bam in parallel

Sample sizes are small for fast run times.

set.seed(3)

dat <- gamSim(1,n=25000,dist="normal",scale=20)

bs <- "cr";k <- 12

b <- bam(y ~ s(x0,bs=bs)+s(x1,bs=bs)+s(x2,bs=bs,k=k)+
 s(x3,bs=bs),data=dat)

summary(b)

plot(b,pages=1,rug=FALSE) ## plot smooths, but not rug

plot(b,pages=1,rug=FALSE,seWithMean=TRUE) ## `with intercept' CIs

ba <- bam(y ~ s(x0,bs=bs,k=k)+s(x1,bs=bs,k=k)+s(x2,bs=bs,k=k)+
 s(x3,bs=bs,k=k),data=dat,method="GCV.Cp") ## use GCV

summary(ba)

A Poisson example...

k <- 15

dat <- gamSim(1,n=21000,dist="poisson",scale=.1)

system.time(b1 <- bam(y ~ s(x0,bs=bs)+s(x1,bs=bs)+s(x2,bs=bs,k=k),
 data=dat,family=poisson()))

b1

```
## Similar using faster discrete method...
```

```
system.time(b2 <- bam(y ~ s(x0,bs=bs,k=k)+s(x1,bs=bs,k=k)+s(x2,bs=bs,k=k)+  
  s(x3,bs=bs,k=k),data=dat,family=poisson(),discrete=TRUE))  
b2
```

bam.update

[bam](#)

```
bam.update(b,data,chunk.size=10000)
```

```
b          gambamgaussianidentity  
data       bweightsAR.startAR.startNULL  
chunk.size
```

```
bam.updatebrhobam
```

```
"gam"gamObject
```

```
<simon.wood@r-project.org>
```

[mgcv-packagebam](#)

```
library(mgcv)  
## following is not *very* large, for obvious reasons...  
set.seed(8)  
n <- 5000  
dat <- gamSim(1,n=n,dist="normal",scale=5)  
dat[c(50,13,3000,3005,3100),]<- NA  
dat1 <- dat[(n-999):n,]  
dat0 <- dat[1:(n-1000),]
```

```

bs <- "ps";k <- 20
method <- "GCV.Cp"
b <- bam(y ~ s(x0,bs=bs,k=k)+s(x1,bs=bs,k=k)+s(x2,bs=bs,k=k)+
          s(x3,bs=bs,k=k),data=dat0,method=method)

b1 <- bam.update(b,dat1)

b2 <- bam.update(bam.update(b,dat1[1:500,]),dat1[501:1000,])

b3 <- bam(y ~ s(x0,bs=bs,k=k)+s(x1,bs=bs,k=k)+s(x2,bs=bs,k=k)+
          s(x3,bs=bs,k=k),data=dat,method=method)
b1;b2;b3

## example with AR1 errors...

e <- rnorm(n)
for (i in 2:n) e[i] <- e[i-1]*.7 + e[i]
dat$y <- dat$f + e*3
dat[c(50,13,3000,3005,3100),]<- NA
dat1 <- dat[(n-999):n,]
dat0 <- dat[1:(n-1000),]

b <- bam(y ~ s(x0,bs=bs,k=k)+s(x1,bs=bs,k=k)+s(x2,bs=bs,k=k)+
          s(x3,bs=bs,k=k),data=dat0,rho=0.7)

b1 <- bam.update(b,dat1)

summary(b1);summary(b2);summary(b3)

```

bandchol

A

bandchol(B)

B AA

$\text{dpbtrfLAPACK} O(k^2 n) O(n^3)$

$\text{Rt}(\text{R})\%*\% \text{R} = \text{ARBARBBR}$

<simon.wood@r-project.org>

```
require(mgcv)
## simulate a banded diagonal matrix
n <- 7;set.seed(8)
A <- matrix(0,n,n)
sdiag(A) <- runif(n);sdiag(A,1) <- runif(n-1)
sdiag(A,2) <- runif(n-2)
A <- crossprod(A)

## full matrix form...
bandchol(A)
chol(A) ## for comparison

## compact storage form...
B <- matrix(0,3,n)
B[1,] <- sdiag(A);B[2,1:(n-1)] <- sdiag(A,1)
B[3,1:(n-2)] <- sdiag(A,2)
bandchol(B)
```

bcg

`gambam` $y > 0$

$$z = \begin{cases} (y^\lambda - 1)/\lambda & \lambda \neq 0 \\ \log(y) & \lambda = 0 \end{cases}$$
$$\mu w^{-1/2} \exp(\theta) w^{1/2} (z - \mu) \exp(-\theta) N(0, 1)$$
$$z \sim N(\mu, e^{2\theta} w^{-1}).$$

$\theta \lambda$

```
bcg(theta=NULL,link="identity")
```

```
theta      λθλexp(θ)
link       "identity""log""sqrt"
```

y
 ∂y
 $\text{Inf} y$

extended.family

<simon.wood@r-project.org>

[cnormcpoisclog](#)

```
library(mgcv)

set.seed(3) ## Simulate some gamma data?
dat <- gamSim(1,n=400,dist="normal",scale=1)
dat$f <- dat$f/4 ## true linear predictor
Ey <- exp(dat$f);scale <- .5 ## mean and GLM scale parameter
dat$y <- rgamma(Ey*0,shape=1/scale,scale=Ey*scale)

## Just Box-Cox no censoring...
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=bcg,data=dat)
summary(b)
plot(b,pages=1,scheme=1)

## try various censoring...
yb <- cbind(dat$y,dat$y)
ind <- 1:100
yb[ind,2] <- yb[ind,1] + runif(100)*3
yb[51:100,2] <- 0 ## left censored
yb[101:140,2] <- Inf ## right censored

b <- gam(yb~s(x0)+s(x1)+s(x2)+s(x3),family=bcg,data=dat)
summary(b)
plot(b,pages=1,scheme=1)
```

betar

[gambam](#) $\mu\mu(1-\mu)/(1+\phi)\phi$

betar(theta = NULL, link = "logit",eps=.Machine\$double.eps*100)

theta	ϕ
link	"logit""probit""cloglog""cauchit"
eps	[eps,1-eps]eps

$\text{eps} < \text{epseps} > 1 - \text{eps} \mu \phi > 1 \text{epseps} (1 - \mu) \phi > 1$

extended.family

```
library(mgcv)
## Simulate some beta data...
set.seed(3); n<-400
dat <- gamSim(1,n=n)
mu <- binomial()$linkinv(dat$f/4-2)
phi <- .5
a <- mu*phi; b <- phi - a;
dat$y <- rbeta(n,a,b)

bm <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=betar(link="logit"),data=dat)

bm
plot(bm,pages=1)
```

blas.thread.test

nthreads>1

blas.thread.test(n=1000,nt=4)

n

nt

USE_LOCKING=1

nthreads**bamgambam**(...,discrete=TRUE)

MKL_NUM_THREADS=1

<simon.wood@r-project.org>

bug.reports.mgcv

mgcvmgcv
<simon.wood@r-project.org>mgcv

[sessionInfo](#)

loadread.table

<simon.wood@r-project.org>

choldrop

$\text{RAcholdropA}[-k, -k] \text{kcholup} A + uu^T A - uu^T$

choldrop(R,k)
cholup(R,u,up)

R	A
k	A
u	
up	TRUE

$\text{choldropRt}(R[-k, -k]) \%*\% R[-k, -k] == A[-k, -k] R[-k, -k] A[-k, -k] R n O(n^2)$
 $\text{cholup} R R^T R + uu^T = [u, R^T] [u, R^T]^T = [u, R^T] Q^T Q [u, R^T]^T Q Q [u, R^T]^T = [0, R_1^T]^T R_1 Q u$
 $A - uu^T O(n^2)$

<simon.wood@r-project.org>

```

require(mgcv)
set.seed(0)
n <- 6
A <- crossprod(matrix(runif(n*n),n,n))
R0 <- chol(A)
k <- 3
Rd <- choldrop(R0,k)
range(Rd-chol(A[-k,-k]))
Rd;chol(A[-k,-k])

## same but using lower triangular factor A = LL'
L <- t(R0)
Ld <- choldrop(L,k)
range(Ld-t(chol(A[-k,-k])))
Ld;t(chol(A[-k,-k]))

## Rank one update example
u <- runif(n)
R <- cholup(R0,u,TRUE)
Ru <- chol(A+u %*% t(u)) ## direct for comparison
R;Ru
range(R-Ru)

## Downdate - just going back from R to R0
Rd <- cholup(R,u,FALSE)
R0;Rd
range(R0-Rd)

```

choose.k

```

kmgcvstek
k-1kstekfx=TRUE
k
kk-1k-1kk
kkk
gam.checkkkplot.gam
k=10k=20k=20k=10

```

<simon.wood@r-project.org>


```

## Simulate some data ....
library(mgcv)
set.seed(1)
dat <- gamSim(1,n=400,scale=2)

## fit a GAM with quite low `k`
b<-gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=6)+s(x3,k=6),data=dat)
plot(b,pages=1,residuals=TRUE) ## hint of a problem in s(x2)

## the following suggests a problem with s(x2)
gam.check(b)

## Another approach (see below for more obvious method)....
## check for residual pattern, removeable by increasing `k`
## typically `k`, below, should be substantially larger than
## the original, `k` but certainly less than n/2.
## Note use of cheap "cs" shrinkage smoothers, and gamma=1.4
## to reduce chance of overfitting...
rsd <- residuals(b)
gam(rsd~s(x0,k=40,bs="cs"),gamma=1.4,data=dat) ## fine
gam(rsd~s(x1,k=40,bs="cs"),gamma=1.4,data=dat) ## fine
gam(rsd~s(x2,k=40,bs="cs"),gamma=1.4,data=dat) ## `k` too low
gam(rsd~s(x3,k=40,bs="cs"),gamma=1.4,data=dat) ## fine

## refit...
b <- gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=20)+s(x3,k=6),data=dat)
gam.check(b) ## better

## similar example with multi-dimensional smooth
b1 <- gam(y~s(x0)+s(x1,x2,k=15)+s(x3),data=dat)
rsd <- residuals(b1)
gam(rsd~s(x0,k=40,bs="cs"),gamma=1.4,data=dat) ## fine
gam(rsd~s(x1,x2,k=100,bs="ts"),gamma=1.4,data=dat) ## `k` too low
gam(rsd~s(x3,k=40,bs="cs"),gamma=1.4,data=dat) ## fine

gam.check(b1) ## shows same problem

## and a `te` example
b2 <- gam(y~s(x0)+te(x1,x2,k=4)+s(x3),data=dat)
rsd <- residuals(b2)
gam(rsd~s(x0,k=40,bs="cs"),gamma=1.4,data=dat) ## fine
gam(rsd~te(x1,x2,k=10,bs="cs"),gamma=1.4,data=dat) ## `k` too low
gam(rsd~s(x3,k=40,bs="cs"),gamma=1.4,data=dat) ## fine

gam.check(b2) ## shows same problem

## same approach works with other families in the original model
dat <- gamSim(1,n=400,scale=.25,dist="poisson")
bp<-gam(y~s(x0,k=5)+s(x1,k=5)+s(x2,k=5)+s(x3,k=5),
        family=poisson,data=dat,method="ML")

gam.check(bp)

rsd <- residuals(bp)
gam(rsd~s(x0,k=40,bs="cs"),gamma=1.4,data=dat) ## fine
gam(rsd~s(x1,k=40,bs="cs"),gamma=1.4,data=dat) ## fine

```

```
gam(rsd~s(x2,k=40,bs="cs"),gamma=1.4,data=dat) ## `k' too low
gam(rsd~s(x3,k=40,bs="cs"),gamma=1.4,data=dat) ## fine

rm(dat)

## More obvious, but more expensive tactic... Just increase
## suspicious k until fit is stable.

set.seed(0)
dat <- gamSim(1,n=400,scale=2)
## fit a GAM with quite low `k'
b <- gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=6)+s(x3,k=6),
         data=dat,method="REML")
b
## edf for 3rd smooth is highest as proportion of k -- increase k
b <- gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=12)+s(x3,k=6),
         data=dat,method="REML")
b
## edf substantially up, -ve REML substantially down
b <- gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=24)+s(x3,k=6),
         data=dat,method="REML")
b
## slight edf increase and -ve REML change
b <- gam(y~s(x0,k=6)+s(x1,k=6)+s(x2,k=40)+s(x3,k=6),
         data=dat,method="REML")
b
## definitely stabilized (but really k around 20 would have been fine)
```

clog

gambay $\mu s = w^{-1/2} \exp(\theta) y$

$$\frac{\exp\{-(y - \mu)/s\}}{s[1 + \exp\{-(y - \mu)/s\}]^2}.$$

θ

clog(theta=NULL,link="identity")

theta

link "identity""log""sqrt"

y

$-\text{Inf}y$

$\text{Inf}y$

extended.family

```
library(mgcv)

#####
## AFT model example for colon cancer survival data...
#####

library(survival) ## for data
col1 <- colon[colon$type==1,] ## concentrate on single event
col1$differ <- as.factor(col1$differ)
col1$sex <- as.factor(col1$sex)

## set up the AFT response...
logt <- cbind(log(col1$time),log(col1$time))
logt[col1$status==0,2] <- Inf ## right censoring
col1$logt <- -logt ## -ve conventional for AFT versus Cox PH comparison

## fit the model...
b <- gam(logt~s(age,by=sex)+sex+s(nodes)+perfor+rx+obstruct+adhere,
         family=clog(),data=col1)
plot(b,pages=1)
## ... compare this to ?cox.ph
```

cnorm

$$\text{gambay} \mu w^{-1/2} \exp(\theta) w^{1/2} (y - \mu) \exp(-\theta) N(0, 1)$$
$$y \sim N(\mu, e^{2\theta} w^{-1}).$$
$$\theta$$

cnorm(theta=NULL,link="identity")

theta

link "identity" "log" "sqrt"

y
 $-\text{Inf}y$
 $\text{Inf}y$

extended.family

<simon.wood@r-project.org>

```
library(mgcv)

#####
## AFT model example for colon cancer survival data...
#####

library(survival) ## for data
col1 <- colon[colon$etype==1,] ## concentrate on single event
col1$differ <- as.factor(col1$differ)
col1$sex <- as.factor(col1$sex)

## set up the AFT response...
logt <- cbind(log(col1$time),log(col1$time))
logt[col1$status==0,2] <- Inf ## right censoring
col1$logt <- -logt ## -ve conventional for AFT versus Cox PH comparison

## fit the model...
b <- gam(logt~s(age,by=sex)+sex+s(nodes)+perfor+rx+obstruct+adhere,
         family=cnorm(),data=col1)
plot(b,pages=1)
## ... compare this to ?cox.ph

#####
## A Tobit regression example...
#####

set.seed(3);n<-400
dat <- gamSim(1,n=n)
ys <- dat$y - 5 ## shift data down

## truncate at zero, and set up response indicating this has happened...
y <- cbind(ys,ys)
y[ys<0,2] <- -Inf
y[ys<0,1] <- 0
```

```

dat$yt <- y
b <- gam(yt~s(x0)+s(x1)+s(x2)+s(x3),family=cnorm,data=dat)
plot(b,pages=1)

#####
## A model for rounded data...
#####

dat <- gamSim(1,n=n)
y <- round(dat$y)
y <- cbind(y-.5,y+.5) ## set up to indicate interval censoring
dat$yi <- y
b <- gam(yi~s(x0)+s(x1)+s(x2)+s(x3),family=cnorm,data=dat)
plot(b,pages=1)

```

columb

```

data(columb)
data(columb.polys)

```

```

columb

```

```

names(columb.polys)
columb.polys

```

```

columbcolumb.polyscolumb.polys[[i]]columb.polys[[2]]columb.polys[[1]]
names(columb.polys)columb$district

```

```

columbusspdep

```

```

## see ?mrf help files

```

concurvity

gam

concurvity(b,full=TRUE)

b	"gam"
full	TRUEFALSE

full=TRUEfull=FALSE

<simon.wood@r-project.org>

```
library(mgcv)
## simulate data with concurvity...
set.seed(8);n<- 200
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10
t <- sort(runif(n)) ## first covariate
## make covariate x a smooth function of t + noise...
x <- f2(t) + rnorm(n)*3
## simulate response dependent on t and x...
y <- sin(4*pi*t) + exp(x/20) + rnorm(n)*.3

## fit model...
b <- gam(y ~ s(t,k=15) + s(x,k=15),method="REML")

## assess concurvity between each term and `rest of model'...
concurvity(b)

## ... and now look at pairwise concurvity between terms...
concurvity(b,full=FALSE)
```

cox.ph

cox.phgamweightscox.phcox.pht

cox.ph(link="identity")

link "identity"

gamweightsgam

predicttype="response"fitted.values
deviancemartingalescoreschoenfeld"terms"

bam(...,discrete=TRUE)cox.pht

general.family

cox.phtcnorm

```
library(mgcv)
library(survival) ## for data
col1 <- colon[colon$type==1,] ## concentrate on single event
col1$differ <- as.factor(col1$differ)
col1$sex <- as.factor(col1$sex)

b <- gam(time~s(age,by=sex)+sex+s(nodes)+perfor+rx+obstruct+adhere,
          family=cox.ph(),data=col1,weights=status)

summary(b)

plot(b,pages=1,all.terms=TRUE) ## plot effects

plot(b$linear.predictors,residuals(b))
```

```

## plot survival function for patient j...

np <- 300; j <- 6
newd <- data.frame(time=seq(0,3000,length=np))
dname <- names(col1)
for (n in dname) newd[[n]] <- rep(col1[[n]][j],np)
newd$time <- seq(0,3000,length=np)
fv <- predict(b,newdata=newd,type="response",se=TRUE)
plot(newd$time,fv$fit,type="l",ylim=c(0,1),xlab="time",ylab="survival")
lines(newd$time,fv$fit+2*fv$se.fit,col=2)
lines(newd$time,fv$fit-2*fv$se.fit,col=2)

## crude plot of baseline survival...

plot(b$family$data$str,exp(-b$family$data$h),type="l",ylim=c(0,1),
     xlab="time",ylab="survival")
lines(b$family$data$str,exp(-b$family$data$h + 2*b$family$data$q^.5),col=2)
lines(b$family$data$str,exp(-b$family$data$h - 2*b$family$data$q^.5),col=2)
lines(b$family$data$str,exp(-b$family$data$km),lty=2) ## Kaplan Meier

## Checking the proportional hazards assumption via scaled score plots as
## in Klein and Moeschberger Section 11.4 p374-376...

ph.resid <- function(b,stratum=1) {
  ## convenience function to plot scaled score residuals against time,
  ## by term. Reference lines at 5% exceedance prob for Brownian bridge
  ## (see KS test statistic distribution).
  rs <- residuals(b,"score"); term <- attr(rs,"term")
  if (is.matrix(b$y)) {
    ii <- b$y[,2] == stratum; b$y <- b$y[ii,1]; rs <- rs[ii,]
  }
  oy <- order(b$y)
  for (i in 1:length(term)) {
    ii <- term[[i]]; m <- length(ii)
    plot(b$y[oy],rs[oy,ii[1]],ylim=c(-3,3),type="l",ylab="score residuals",
         xlab="time",main=names(term)[i])
    if (m>1) for (k in 2:m) lines(b$y[oy],rs[oy,ii[k]],col=k);
    abline(-1.3581,0,lty=2); abline(1.3581,0,lty=2)
  }
}
par(mfrow=c(2,2))
ph.resid(b)

## stratification example, with 2 randomly allocated strata
## so that results should be similar to previous...
col1$strata <- sample(1:2,nrow(col1),replace=TRUE)
bs <- gam(cbind(time,strata)~s(age,by=sex)+sex+s(nodes)+perfor+rx+obstruct
          +adhere,family=cox.ph(),data=col1,weights=status)
plot(bs,pages=1,all.terms=TRUE) ## plot effects

## baseline survival plots by strata...

for (i in 1:2) { ## loop over strata
  ## create index selecting elements of stored hazard info for stratum i...
  ind <- which(bs$family$data$strat == i)
  if (i==1) plot(bs$family$data$str[ind],exp(-bs$family$data$h[ind]),type="l",

```



```

ylim=c(0,1),xlab="time",ylab="survival",lwd=2,col=i) else
  lines(bs$family$data$tr[ind],exp(-bs$family$data$h[ind]),lwd=2,col=i)
lines(bs$family$data$tr[ind],exp(-bs$family$data$h[ind] +
  2*bs$family$data$q[ind]^0.5),lty=2,col=i) ## upper ci
lines(bs$family$data$tr[ind],exp(-bs$family$data$h[ind] -
  2*bs$family$data$q[ind]^0.5),lty=2,col=i) ## lower ci
lines(bs$family$data$tr[ind],exp(-bs$family$data$km[ind]),col=i) ## KM
}

```

Simple simulated known truth example...

```

ph.weibull.sim <- function(eta,gamma=1,h0=.01,t1=100) {
  lambda <- h0*exp(eta)
  n <- length(eta)
  U <- runif(n)
  t <- (-log(U)/lambda)^(1/gamma)
  d <- as.numeric(t <= t1)
  t[!d] <- t1
  list(t=t,d=d)
}

n <- 500;set.seed(2)
x0 <- runif(n, 0, 1);x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1);x3 <- runif(n, 0, 1)
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
f3 <- function(x) 0*x
f <- f0(x0) + f1(x1) + f2(x2)
g <- (f-mean(f))/5
surv <- ph.weibull.sim(g)
surv$x0 <- x0;surv$x1 <- x1;surv$x2 <- x2;surv$x3 <- x3

b <- gam(t~s(x0)+s(x1)+s(x2,k=15)+s(x3),family=cox.ph,weights=d,data=surv)

plot(b,pages=1)

## Another one, including a violation of proportional hazards for
## effect of x2...

set.seed(2)
h <- exp((f0(x0)+f1(x1)+f2(x2)-10)/5)
t <- rexp(n,h);d <- as.numeric(t<20)

## first with no violation of PH in the simulation...
b <- gam(t~s(x0)+s(x1)+s(x2)+s(x3),family=cox.ph,weights=d)
plot(b,pages=1)
ph.resid(b) ## fine

## Now violate PH for x2 in the simulation...
ii <- t>1.5
h1 <- exp((f0(x0)+f1(x1)+3*f2(x2)-10)/5)
t[ii] <- 1.5 + rexp(sum(ii),h1[ii]);d <- as.numeric(t<20)

b <- gam(t~s(x0)+s(x1)+s(x2)+s(x3),family=cox.ph,weights=d)
plot(b,pages=1)
ph.resid(b) ## The checking plot picks up the problem in s(x2)

```

```
## conditional logistic regression models are often estimated using the
## cox proportional hazards partial likelihood with a strata for each
## case-control group. A dummy vector of times is created (all equal).
## The following compares to 'clogit' for a simple case. Note that
## the gam log likelihood is not exact if there is more than one case
## per stratum, corresponding to clogit's approximate method.
library(survival);library(mgcv)
infert$dumt <- rep(1,nrow(infert))
mg <- gam(cbind(dumt,stratum) ~ spontaneous + induced, data=infert,
         family=cox.ph,weights=case)
ms <- clogit(case ~ spontaneous + induced + strata(stratum), data=infert,
            method="approximate")
summary(mg)$p.table[1:2,]; ms
```

cox.pht

cox.ph

```
pbcseqsurvivalbamdiscrete=TRUEcox.phgammethod="REML"
tdpois
```

```
require(mgcv);require(survival)

## First define functions for producing Poisson model data frame

app <- function(x,t,to) {
  ## wrapper to approx for calling from apply...
  y <- if (sum(!is.na(x))<1) rep(NA,length(to)) else
    approx(t,x,to,method="constant",rule=2)$y
  if (is.factor(x)) factor(levels(x)[y],levels=levels(x)) else y
} ## app

tdpois <- function(dat,event="z",et="fuptime",t="day",status="status1",
                  id="id") {
  ## dat is data frame. id is patient id; et is event time; t is
  ## observation time; status is 1 for death 0 otherwise;
  ## event is name for Poisson response.
  if (event %in% names(dat)) warning("event name in use")
  require(utils) ## for progress bar
  te <- sort(unique(dat[[et]][dat[[status]]==1])) ## event times
  sid <- unique(dat[[id]])
  inter <- interactive()
  if (inter) prg <- txtProgressBar(min = 0, max = length(sid), initial = 0,
    char = "=",width = NA, title="Progress", style = 3)
  ## create dataframe for poisson model data
  dat[[event]] <- 0; start <- 1
```

```

dap <- dat[rep(1:length(sid),length(te)),]
for (i in 1:length(sid)) { ## work through patients
  di <- dat[dat[[id]]==sid[i],] ## ith patient's data
  tr <- te[te <= di[[et]][1]] ## times required for this patient
  ## Now do the interpolation of covariates to event times...
  um <- data.frame(lapply(X=di,FUN=app,t=di[[t]],to=tr))
  ## Mark the actual event...
  if (um[[et]][1]==max(tr)&&um[[status]][1]==1) um[[event]][nrow(um)] <- 1
  um[[et]] <- tr ## reset time to relevant event times
  dap[start:(start-1+nrow(um)),] <- um ## copy to dap
  start <- start + nrow(um)
  if (inter) setTxtProgressBar(prg, i)
}
if (inter) close(prg)
dap[1:(start-1),]
} ## tdpois

## The following typically takes a minute or less...

## Convert pbcseq to equivalent Poisson form...
pbcseq$status1 <- as.numeric(pbcseq$status==2) ## death indicator
pb <- tdpois(pbcseq) ## conversion
pb$tf <- factor(pb$futime) ## add factor for event time

## Fit Poisson model...
b <- bam(z ~ tf - 1 + sex + trt + s(sqrt(protime)) + s(platelet)+ s(age)+
s(bili)+s(albumin), family=poisson,data=pb,discrete=TRUE,nthreads=2)

pb$dumt <- rep(1,nrow(pb)) ## dummy time
## Fit as conditional logistic...
b1 <- gam(cbind(dumt,tf) ~ sex + trt + s(sqrt(protime)) + s(platelet)
+ s(age) + s(bili) + s(albumin),family=cox.ph,data=pb,weights=z)

par(mfrow=c(2,3))
plot(b,scale=0)

## compute residuals...
chaz <- tapply(fitted(b),pb$id,sum) ## cum haz by subject
d <- tapply(pb$z,pb$id,sum) ## censoring indicator
mrds <- d - chaz ## Martingale
drsd <- sign(mrds)*sqrt(-2*(mrds + d*log(chaz))) ## deviance

## plot survivor function and s.e. band for subject 25
te <- sort(unique(pb$futime)) ## event times
di <- pbcseq[pbcseq$id==25,] ## data for subject 25
pd <- data.frame(lapply(X=di,FUN=app,t=di$day,to=te)) ## interpolate to te
pd$tf <- factor(te)
X <- predict(b,newdata=pd,type="lpmatrix")
eta <- drop(X%*%coef(b)); H <- cumsum(exp(eta))
J <- apply(exp(eta)*X,2,cumsum)
se <- diag(J%*%vcov(b)%*%t(J))^0.5
plot(stepfun(te,c(1,exp(-H))),do.points=FALSE,ylim=c(0.7,1),
      ylab="S(t)",xlab="t (days)",main="",lwd=2)
lines(stepfun(te,c(1,exp(-H+se))),do.points=FALSE)
lines(stepfun(te,c(1,exp(-H-se))),do.points=FALSE)
rug(pbcseq$day[pbcseq$id==25]) ## measurement times

```

cpois

[gambam](#)

```
cpois(link="log")
```

```
link          "identity""log""sqrt"
```

```
  y
  -Infy
  Infy
y(2,3)
```

```
extended.family
```

```
<simon.wood@r-project.org>
```

```
library(mgcv)
set.seed(6); n <- 2000
dat <- gamSim(1,n=n,dist="poisson",scale=.1) ## simulate Poi data

## illustration that cpois an poisson give same results if there
## is no censoring...

b0 <- gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+
          s(x3,bs="cr"),family=poisson,data=dat,method="REML")
plot(b0,pages=1,scheme=2)

b1 <- gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+
          s(x3,bs="cr"),family=cpois,data=dat)
plot(b1,pages=1,scheme=2)

b0;b1
```

```

## Now censor some observations...
dat1 <- dat
m <- 300
y <- matrix(dat$y,n,ncol=2) ## response matrix
ii <- sample(n,3*m) ## censor these
for (i in 1:m) { ## right, left, interval...
  j <- ii[i]; if (y[j,1] > 4.5) y[j,] <- c(4.5,Inf)
  j <- ii[m+i]; if (y[j,1] < 2.5) y[j,] <- c(2.5,-Inf)
  j <- ii[2*m+i]; if (y[j,1] > 1.5 & y[j,1] < 5.5) y[j,] <- c(1.5,5.5)
}
n - sum(y[,1]==y[,2]) ## number of censored obs
dat1$y <- y

## now fit model with censoring...
b2 <- gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+
  s(x3,bs="cr"),family=cpois,data=dat1)
plot(b2,pages=1,scheme=2);b2

```

cSplineDes

splineDesign

cSplineDes(x, knots, ord = 4, derivs=0)

x

knots

ord

derivs ordx

length(x)length(knots)-1

<simon.wood@r-project.org>

[cyclic.p.spline](#)

```

require(mgcv)
## create some x's and knots...
n <- 200
x <- 0:(n-1)/(n-1);k<- 0:5/5
X <- cSplineDes(x,k) ## cyclic spline design matrix
## plot evaluated basis functions...
plot(x,X[,1],type="l"); for (i in 2:5) lines(x,X[,i],col=i)
## check that the ends match up...
ee <- X[1,]-X[n,];ee
tol <- .Machine$double.eps^.75
if (all.equal(ee,ee*0,tolerance=tol)!=TRUE)
  stop("cyclic spline ends don't match!")

## similar with uneven data spacing...
x <- sort(runif(n)) + 1 ## sorting just makes end checking easy
k <- seq(min(x),max(x),length=8) ## create knots
X <- cSplineDes(x,k) ## get cyclic spline model matrix
plot(x,X[,1],type="l"); for (i in 2:ncol(X)) lines(x,X[,i],col=i)
ee <- X[1,]-X[n,];ee ## do ends match??
tol <- .Machine$double.eps^.75
if (all.equal(ee,ee*0,tolerance=tol)!=TRUE)
  stop("cyclic spline ends don't match!")

```

dDeta

```
mu = inv_link(eta)
```

```
dDeta(y, mu, wt, theta, fam, deriv = 0)
```

```

y
mu          etamu = inv_link(eta)mu=E(y)
wt
theta
fam
deriv

```

dpnorm

$N(0, 1)$

dpnorm(x0, x1, log.p=FALSE)

x0

x1

log.p TRUE

pnorm(x1)-pnorm(x0)x0x1

<simon.wood@r-project.org>

```
require(mgcv)
x <- seq(-10,10,length=10000)
eps <- 1e-10
y0 <- pnorm(x+eps)-pnorm(x) ## cancellation prone
y1 <- dpnorm(x,x+eps)        ## stable
## illustrate stable computation in black, and
## cancellation prone in red...
par(mfrow=c(1,2),mar=c(4,4,1,1))
plot(log(y1),log(y0),type="l")
lines(log(y1[x>0]),log(y0[x>0]),col=2)
plot(x,log(y1),type="l")
lines(x,log(y0),col=2)
```

dppois

dppois(y0,y1,mu,log.p=TRUE)

y0

y1

mu

log.p FALSE

```
log(ppois(y1,mu)-ppois(y0,mu))y0y1log(0)
```

```
<simon.wood@r-project.org>
```

```
require(mgcv)
n <- 10
mu <- rep(2,n)
y0 <- c(0:5,10,23,0,2)
y1 <- c(1:6,11,1000,100,40)
dppois(y0,y1,mu)
log(ppois(y1,mu)-ppois(y0,mu))
```

```
exclude.too.far
```

```
TRUEFALSEvis.gamplot.gam
```

```
exclude.too.far(g1,g2,d1,d2,dist)
```

```
g1
g2
d1
d2
dist          dist
```

```
g1g2g1g2d1d2distTRUEFALSE
```

```
TRUEg1g2
```

```
<simon.wood@r-project.org>
```

```
vis.gam
```

```
library(mgcv)
x<-rnorm(100);y<-rnorm(100) # some "data"
n<-40 # generate a grid...
mx<-seq(min(x),max(x),length=n)
my<-seq(min(y),max(y),length=n)
gx<-rep(mx,n);gy<-rep(my,rep(n,n))
tf<-exclude.too.far(gx,gy,x,y,0.1)
plot(gx[!tf],gy[!tf],pch=".");points(x,y,col=2)
```

`extract.lme.cov`

[gamm](#)`lmeextract.lme.covextract.lme.cov2`

```
extract.lme.cov(b,data=NULL,start.level=1)
extract.lme.cov2(b,data=NULL,start.level=1)
```

`b` [lme](#)

`data` [lme](#)
`start.level`

```
extract.lme.covextract.lme.cov2
gammextract.lme.cov2
extract.lme.covextract.lme.cov2
```

```
extract.lme.cov
extract.lme.cov2
```

`<simon.wood@r-project.org>`

`lme`

`lme`

[gammformXtViX](#)

```
## see also ?formXtViX for use of extract.lme.cov2
require(mgcv)
library(nlme)
data(Rail)
b <- lme(travel~1,Rail,~1|Rail)
extract.lme.cov(b)
extract.lme.cov2(b)
```

factor.smooth

```
by bystet2ids(x,by=fac,id=1)xfac
bs="sz"s(...,bs="sz")bysmooth.construct.sz.smooth.spec
bs="fs"sz"gammlmesmooth.construct.fs.smooth.spec
```

<simon.wood@r-project.org>

[smooth.construct.fs.smooth.spec](#)[smooth.construct.sz.smooth.spec](#)

```
library(mgcv)
set.seed(0)
## simulate data...
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x,a=2,b=-1) exp(a * x)+b
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
      (10 * x)^3 * (1 - x)^10
n <- 500;nf <- 25
fac <- sample(1:nf,n,replace=TRUE)
x0 <- runif(n);x1 <- runif(n);x2 <- runif(n)
a <- rnorm(nf)*.2 + 2;b <- rnorm(nf)*.5
f <- f0(x0) + f1(x1,a[fac],b[fac]) + f2(x2)
fac <- factor(fac)
y <- f + rnorm(n)*2
## so response depends on global smooths of x0 and
## x2, and a smooth of x1 for each level of fac.

## fit model...
bm <- gamm(y~s(x0)+ s(x1,fac,bs="fs",k=5)+s(x2,k=20))
plot(bm$gam,pages=1)
summary(bm$gam)

bd <- bam(y~s(x0)+ s(x1) + s(x1,fac,bs="sz",k=5)+s(x2,k=20),discrete=TRUE)
plot(bd,pages=1)
summary(bd)

## Could also use...
## b <- gamm(y~s(x0)+ s(x1,fac,bs="fs",k=5)+s(x2,k=20),method="ML")
## ... but its slower (increasingly so with increasing nf)
## b <- gamm(y~s(x0)+ t2(x1,fac,bs=c("tp","re"),k=5,full=TRUE)+
##       s(x2,k=20),method="ML")
## ... is exactly equivalent.
```

family.mgcv

familyfamilyglmgambamgammgcvgambam

family

[Tweedie](#)pp**tw**pgam/bam

[negbin](#)nbthetagam/bam

extended.familygam"NCV""REML""ML"**bam**

[betar](#)

[cnorm](#)

[bcg](#)

[clog](#)

[cpois](#)

[nbtheta](#)

[ocat](#)

[scat](#)

[tw](#)

[ziP](#)

familyextended.familybinomialnb

[gfam](#)

general.familygam

[cox.ph](#)

[gammals](#)

[gaulss](#)

[gevlss](#)

[gumbls](#)

[multinom](#)

[mvn](#)

[shash](#)

[twlss](#)

[ziplss](#)

FFdes

```
FFdes(size=5,ccd=FALSE)
```

```
size
```

```
ccd          TRUE
```

```
<simon.wood@r-project.org>
```

```
require(mgcv)
plot(rbind(0,FFdes(2,TRUE)),xlab="x",ylab="y",
      col=c(2,1,1,1,1,4,4,4,4),pch=19,main="CCD")
FFdes(5)
FFdes(5,TRUE)
```

fix.family.link

```
fix.family.link(fam)
fix.family.var(fam)
fix.family.ls(fam)
fix.family.qf(fam)
fix.family.rd(fam)
```

```
fam          family
```

```
dvard2vard3vard2linkd3linkd4linklsgamd4linkd3varls
dvarmud2linkmumu
dvard2var
qq.gamqfrdqfrdqf
```

```
dvard2vard2linkd3linkd4linklsqfrdfix.family.varscale
```

```
<simon.wood@r-project.org>
```

```
gam.fit3qq.gam
```

fixDependence

```
X2X1
```

```
fixDependence(X1,X2,tol=.Machine$double.eps^.5,rank.def=0,strict=FALSE)
```

```
X1
X2          X1
tol
rank.def    X2X1tolX2
strict      TRUEX1FALSEX2X1
```

```
X2X1strict==FALSENULL
```

```
<simon.wood@r-project.org>
```

```
library(mgcv)
n<-20;c1<-4;c2<-7
X1<-matrix(runif(n*c1),n,c1)
X2<-matrix(runif(n*c2),n,c2)
X2[,3]<-X1[,2]+X2[,4]*.1
X2[,5]<-X1[,1]*.2+X1[,2]*.04
fixDependence(X1,X2)
fixDependence(X1,X2,strict=TRUE)
```

formula.gam

[gam](#)gam

```
## S3 method for class 'gam'  
formula(x,...)
```

```
x          gamgamObjectgam()  
...
```

[gam](#)mgcv

[gam](#)glmstetit2.gam

```
s(x1,x2,...,k=12,fx=FALSE,bs="tp",by=z,id=1)  
x1x2kkkchoose.kgam.check
```

```
fxkk-1bssmooth.termsuser.defined.smoothbs"tp"tprsbygam(y~s(x,by=z)) $E(y) = f(x)z$   
 $f(\cdot)$ bygam.modelsid
```

```
te(x,z,bs=c("tp","tp"),m=c(2,3),k=c(5,10))  
xztetit2
```

stetit2spspspgamspgam

```
y~s(x)+s(z)+s(x,z)y~s(x,z)+s(z,v)  
tigam.side
```

```
gamby(X,Z,by=L)XZLFXZrowSums(F*L)linear.functional.terms
```

[gam](#)paraPengam.models

[mvn](#)1+3~s(x)+z-1s(x)z-1y ~ -1~ -1[multinom](#)

x\$formulaanova

gamdat[["x"]]

<simon.wood@r-project.org>

[gam](#)

formXtViX

`gammVextract.lme.cov2` $X^T V^{-1} X V$

`formXtViX(V,X)`

V `extract.lme.cov2`
X

`extract.lme.cov2`

`crossprod(R)` $X^T V^{-1} X$

`<simon.wood@r-project.org>`

`lme`

`lme`

`gammextract.lme.cov2`

```
require(mgcv)
library(nlme)
data(ergoStool)
b <- lme(effort ~ Type, data=ergoStool, random=~1|Subject)
V1 <- extract.lme.cov(b, ergoStool)
V2 <- extract.lme.cov2(b, ergoStool)
X <- model.matrix(b, data=ergoStool)
crossprod(formXtViX(V2, X))
t(X)
```

`fs.test`

```
fs.test(x,y,r0=.1,r=.5,l=3,b=1,exclude=TRUE)
fs.boundary(r0=.1,r=.5,l=3,n.theta=20)
```

```
xy
r0
r
l
b
exclude      NA
n.theta
```

```
fs.testNAfs.boundaryx,y
```

```
<simon.wood@r-project.org>
```

```
require(mgcv)
## plot the function, and its boundary...
fsb <- fs.boundary()
m<-300;n<-150
xm <- seq(-1,4,length=m);yn<-seq(-1,1,length=n)
xx <- rep(xm,n);yy<-rep(yn,rep(m,n))
tru <- matrix(fs.test(xx,yy),m,n) ## truth
image(xm,yn,tru,col=heat.colors(100),xlab="x",ylab="y")
lines(fsb$x,fsb$y,lwd=3)
contour(xm,yn,tru,levels=seq(-5,5,by=.25),add=TRUE)
```

gam

[family.mgcvgam](#)

[smooth.termsgam.modelsrandom.effectslinear.functional.termsgam.selection](#)

[gam.checkchoose.k](#)

gam

[bamgamrandom.effects](#)

```
gam(formula,family=gaussian(),data=list(),weights=NULL,subset=NULL,
    na.action,offset=NULL,method="GCV.Cp",
    optimizer=c("outer","newton"),control=list(),scale=0,
    select=FALSE,knots=NULL,sp=NULL,min.sp=NULL,H=NULL,gamma=1,
    fit=TRUE,paraPen=NULL,G=NULL,in.out,drop.unused.levels=TRUE,
    drop.intercept=NULL,nei=NULL,discrete=FALSE,...)
```

formula [formula.gamgam.modelsstetit2](#)

family [glmfamilyfamily.mgcvquasimethod](#)

data environment(formula)gam

weights weights <- weights/mean(weights)

subset

na.action

offset formulalmglm

control [gam.control](#)

method "GCV.Cp""GACV.Cp""NCV"nei"QNCV""REML""P-REML""ML""P-ML""REML"
 "ML""NCV""QNCV"

optimizer method"outer""outer"optimizer"newton""bfgs""optim""nlm""efs"

scale

select TRUEgamgamma

knots [kktprs](#)"tp"/"ts"

sp length(sp)

min.sp full.splength(min.sp)sp

H

gamma gamman/gammaefs

fit TRUEgamFALSEGG

paraPen [gam.models](#)

G NULLgamfit=FALSEspgammain.outscalecontrolmethodoptimizerfit

in.out spscale

drop.unused.levels

```

drop.intercept TRUEformula
nei          NCVamaneis$(nei$ma[j-1]+1):nei$ma[j]]jdmd
             nei$d[(nei$md[j-1]+1):nei$md[j]]nei==NULLamajackknifeTRUEFALSE
             jackknifemethod

discrete     bam

...          gam.fit3mustart

```

$$\log\{E(y_i)\} = \alpha + f_1(x_{1i}) + f_2(x_{2i})$$

$y_i \sim \text{Poi}f_1f_2x_1x_2$ by

f_1f_2

mgcvgam

gammgcv

$$nD/(n - DoF)^2$$

$$D/n + 2sDoF/n - s$$

$DnsDoFs$

NCVmgcv

gamgam.fit3glm.fitbamgammgam.fit3gam

smooth.termssmooth.constructkstetit2k-1kk

gamgamparaPenlinear.functional.termslink{family.mgcv}

gam()gamlostetit2gam

fit=FALSEG

"gam"gamObject

choose.kgam.check

<simon.wood@r-project.org>

<https://webhomes.maths.ed.ac.uk/~swood34/>

`mgcv-package gamObject gam.model smooth.terms linear.functional.terms
predict.gam plot.gam summary.gam gam.side gam.selection gam.control gam.check
linear.functional.terms negbin magic vis.gam`

`## see also examples in ?gam.models (e.g. 'by' variables,
random effects and tricks for large binary datasets)`

```
library(mgcv)
set.seed(2) ## simulate some data...
dat <- gamSim(1,n=400,dist="normal",scale=2)
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
summary(b)
plot(b,pages=1,residuals=TRUE) ## show partial residuals
plot(b,pages=1,seWithMean=TRUE) ## `with intercept' CIs
```

```

## plot derivaives of smooths w.r.t. covariates - the
## direct analogue of linear model regression coefficients
## - also selecting a more colourful plotting scheme...
plot(b,pages=1,scale=0,deriv=TRUE,scheme=2)

## run some basic model checks, including checking
## smoothing basis dimensions...
gam.check(b)

## same fit in two parts .....
G <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),fit=FALSE,data=dat)
b <- gam(G=G)
print(b)

## 2 part fit enabling manipulation of smoothing parameters...
G <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),fit=FALSE,data=dat,sp=b$sp)
G$lsp0 <- log(b$sp*10) ## provide log of required sp vec
gam(G=G) ## it's smoother

## change the smoothness selection method to REML
b0 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="REML")
## use alternative plotting scheme, and way intervals include
## smoothing parameter uncertainty...
plot(b0,pages=1,scheme=1,unconditional=TRUE)

## Would a smooth interaction of x0 and x1 be better?
## Use tensor product smooth of x0 and x1, basis
## dimension 49 (see ?te for details, also ?t2).
bt <- gam(y~te(x0,x1,k=7)+s(x2)+s(x3),data=dat,
          method="REML")
plot(bt,pages=1)
plot(bt,pages=1,scheme=2) ## alternative visualization
AIC(b0,bt) ## interaction worse than additive

## Alternative: test for interaction with a smooth ANOVA
## decomposition (this time between x2 and x1)
bt <- gam(y~s(x0)+s(x1)+s(x2)+s(x3)+ti(x1,x2,k=6),
          data=dat,method="REML")
summary(bt)

## If it is believed that x0 and x1 are naturally on
## the same scale, and should be treated isotropically
## then could try...
bs <- gam(y~s(x0,x1,k=40)+s(x2)+s(x3),data=dat,
          method="REML")
plot(bs,pages=1)
AIC(b0,bt,bs) ## additive still better.

## Now do automatic terms selection as well
b1 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,
          method="REML",select=TRUE)
plot(b1,pages=1)

## set the smoothing parameter for the first term, estimate rest ...
bp <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),sp=c(0.01,-1,-1,-1),data=dat)

```

```

plot(bp,pages=1,scheme=1)
## alternatively...
bp <- gam(y~s(x0,sp=.01)+s(x1)+s(x2)+s(x3),data=dat)

# set lower bounds on smoothing parameters ....
bp<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),
        min.sp=c(0.001,0.01,0,10),data=dat)
print(b);print(bp)

# same with REML
bp<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),
        min.sp=c(0.1,0.1,0,10),data=dat,method="REML")
print(b0);print(bp)

## now a GAM with 3df regression spline term & 2 penalized terms

b0 <- gam(y~s(x0,k=4,fx=TRUE,bs="tp")+s(x1,k=12)+s(x2,k=15),data=dat)
plot(b0,pages=1)

## now simulate poisson data...
set.seed(6)
dat <- gamSim(1,n=2000,dist="poisson",scale=.1)

## use "cr" basis to save time, with 2000 data...
b2<-gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+
        s(x3,bs="cr"),family=poisson,data=dat,method="REML")
plot(b2,pages=1)

## drop x3, but initialize sp's from previous fit, to
## save more time...

b2a<-gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr"),
        family=poisson,data=dat,method="REML",
        in.out=list(sp=b2$sp[1:3],scale=1))
par(mfrow=c(2,2))
plot(b2a)

par(mfrow=c(1,1))
## similar example using GACV...

dat <- gamSim(1,n=400,dist="poisson",scale=.25)

b4<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=poisson,
        data=dat,method="GACV.Cp",scale=-1)
plot(b4,pages=1)

## repeat using REML as in Wood 2011...

b5<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=poisson,
        data=dat,method="REML")
plot(b5,pages=1)

## a binary example (see ?gam.models for large dataset version)...

```

```

dat <- gamSim(1,n=400,dist="binary",scale=.33)

lr.fit <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=binomial,
             data=dat,method="REML")

## plot model components with truth overlaid in red
op <- par(mfrow=c(2,2))
fn <- c("f0","f1","f2","f3");xn <- c("x0","x1","x2","x3")
for (k in 1:4) {
  plot(lr.fit,residuals=TRUE,select=k)
  ff <- dat[[fn[k]]];xx <- dat[[xn[k]]]
  ind <- sort.int(xx,index.return=TRUE)$ix
  lines(xx[ind],(ff-mean(ff))[ind]*.33,col=2)
}
par(op)
anova(lr.fit)
lr.fit1 <- gam(y~s(x0)+s(x1)+s(x2),family=binomial,
              data=dat,method="REML")
lr.fit2 <- gam(y~s(x1)+s(x2),family=binomial,
              data=dat,method="REML")
AIC(lr.fit,lr.fit1,lr.fit2)

## For a Gamma example, see ?summary.gam...

## For inverse Gaussian, see ?rig

## now 2D smoothing...

eg <- gamSim(2,n=500,scale=.1)
attach(eg)

op <- par(mfrow=c(2,2),mar=c(4,4,1,1))

contour(truth$x,truth$z,truth$f) ## contour truth
b4 <- gam(y~s(x,z),data=data) ## fit model
fit1 <- matrix(predict.gam(b4,pr,se=FALSE),40,40)
contour(truth$x,truth$z,fit1) ## contour fit
persp(truth$x,truth$z,truth$f) ## persp truth
vis.gam(b4) ## persp fit
detach(eg)
par(op)

#####
## largish dataset example with user defined knots
#####

par(mfrow=c(2,2))
n <- 5000
eg <- gamSim(2,n=n,scale=.5)
attach(eg)

ind<-sample(1:n,200,replace=FALSE)
b5<-gam(y~s(x,z,k=40),data=data,
        knots=list(x=data$x[ind],z=data$z[ind]))
## various visualizations
vis.gam(b5,theta=30,phi=30)

```

```

plot(b5)
plot(b5,scheme=1,theta=50,phi=20)
plot(b5,scheme=2)

par(mfrow=c(1,1))
## and a pure "knot based" spline of the same data
b6<-gam(y~s(x,z,k=64),data=data,knots=list(x= rep((1:8-0.5)/8,8),
      z=rep((1:8-0.5)/8,rep(8,8))))
vis.gam(b6,color="heat",theta=30,phi=30)

## varying the default large dataset behaviour via `xt`
b7 <- gam(y~s(x,z,k=40,xt=list(max.knots=500,seed=2)),data=data)
vis.gam(b7,theta=30,phi=30)
detach(eg)

```

gam.check

gamgam()gamgamm

```

gam.check(b, old.style=FALSE,
          type=c("deviance", "pearson", "response"),
          k.sample=5000,k.rep=200,
          rep=0, level=.9, r1.col=2, rep.col="gray80", ...)

```

```

b                gamgam()
old.style        TRUE
type             residuals.gam
k.sample
k.rep
replevelr1.colrep.col
                qq.gam()old.style
...

```

gamglmchoose.k

```

k-indexp-valuek.repksamplek.samplekedfNA
kedfkchoose.k
qq.gamqq.gam
gammgamgam

```

<simon.wood@r-project.org>

[choose.kgammagic](#)

```
library(mgcv)
set.seed(0)
dat <- gamSim(1,n=200)
b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b,pages=1)
gam.check(b,pch=19,cex=.3)
```

`gam.control`

`mgcv`[gam](#)methodoptimizer

```
gam.control(nthreads=1,ncv.threads=1,irls.reg=0.0,epsilon = 1e-07,
            maxit = 200,mgcv.tol=1e-7,mgcv.half=15, trace = FALSE,
            rank.tol=.Machine$double.eps^0.5,nlm=list(),
            optim=list(),newton=list(),
            idLinksBases=TRUE,scalePenalty=TRUE,efs.lspmax=15,
            efs.tol=.1,keepData=FALSE,scale.est="fletcher",
            edge.correct=FALSE)
```

nthreads	nthreads
ncv.threads	ncv.threads
irls.reg	irls.reg
epsilon	gam.fit3
maxit	
mgcv.tol	
mgcv.half	
trace	TRUE
rank.tol	
nlm	nlm
optim	optim
newton	


```
idLinksBases      idsFALSEscalePenaltyFALSE
scalePenalty      gammFALSEidLinkBasesFALSE
efs.lspmax
efs.tol
keepData           datagamglm
scale.est           gam.scale
edge.correct       gamTRUEVc
```

```
newtonnewtonconv.tolmaxNstepmaxSstepmaxHalf
nlmnlmnlmnlmndigitsepsilongradtol10*epsilonstepmaxsteptoliterlimcheck.analyticals
FALSE
optimoptimfactr
```

<simon.wood@r-project.org>

[gamglm.control](#)

[gam.convergence](#)

```
gambam
gamgam.outer
bambambam(...,discrete=TRUE)
gammultinomcox.ph
"bfgs"gamoptimizerbambs="cr"stes"tp"
fx=TRUEmaxitgam.controlmin.spgamkste
```

<simon.wood@r-project.org>

`gam.fit`

`mgcvglm.fit`[gammagic](#)[gam](#)[gam.control](#)

```
gam.fit(G, start = NULL, etastart = NULL,
        mustart = NULL, family = gaussian(),
        control = gam.control(), gamma=1,
        fixedSteps=(control$maxit+1),...)
```

```
G           gamfit=FALSE
start
etastart
mustart
family
control     gam.control
gamma
fixedSteps
...
```

<simon.wood@r-project.org>

[gam.fit3](#)[gammagic](#)

gam.fit3

[gam](#)

```
gam.fit3(x, y, sp, Eb ,UrS=list(),
        weights = rep(1, nobs), start = NULL, etastart = NULL,
        mustart = NULL, offset = rep(0, nobs), U1 = diag(ncol(x)),
        Mp = -1, family = gaussian(), control = gam.control(),
        intercept = TRUE,deriv=2,gamma=1,scale=1,
        printWarn=TRUE,scoreType="REML",null.coef=rep(0,ncol(x)),
        pearson.extra=0,dev.extra=0,n.true=-1,S1=NULL,nei=NULL,...)
```

x	
y	
sp	
Eb	
UrS	
weights	
start	
etastart	
mustart	
offset	
U1	
Mp	
family	gaussian()
control	glm.control
intercept	TRUEFALSE
deriv	
gamma	
scale	
printWarn	FALSE
scoreType	
null.coef	
pearson.extra	
dev.extra	
n.true	X
S1	
nei	gam
...	

[glm.fit](#)

[fix.family.link](#)

<simon.wood@r-project.org>
[glm.fit](#)[glm.fit](#)

[gammagic](#)

[gam.fit5.post.proc](#)

[gam.fit5](#)

[gam.fit5.post.proc\(object, Sl, L, lsp0, S, off, gamma\)](#)

object	gam.fit5
Sl	Sl.setup
L	
lsp0	
S	
off	
gamma	

R
Vb
VeVb
Vc
F
[edfdiag\(F\)](#)
[edf2diag\(2F-FF\)](#)

gam.mh

[ginla](#)

```
gam.mh(b,ns=10000,burn=1000,t.df=40,rw.scale=.25,thin=1)
```

```
b          gambam
ns
burn       ns
t.df
rw.scale
thin       thin
```

```
gam.mh
rw.step
```

```
bs
```

```
<simon.wood@r-project.org>
```

```
library(mgcv)
set.seed(3);n <- 400

#####
## First example: simulated Tweedie model...
#####

dat <- gamSim(1,n=n,dist="poisson",scale=.2)
dat$y <- rTweedie(exp(dat$f),p=1.3,phi=.5) ## Tweedie response
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=tw(),
        data=dat,method="REML")

## simulate directly from Gaussian approximate posterior...
br <- rmvn(1000,coef(b),vcov(b))

## Alternatively use MH sampling...
br <- gam.mh(b,thin=2,ns=2000,rw.scale=.15)$bs
```

```

## If 'coda' installed, can check effective sample size
## require(coda);effectiveSize(as.mcmc(br))

## Now compare simulation results and Gaussian approximation for
## smooth term confidence intervals...
x <- seq(0,1,length=100)
pd <- data.frame(x0=x,x1=x,x2=x,x3=x)
X <- predict(b,newdata=pd,type="lpmatrix")
par(mfrow=c(2,2))
for(i in 1:4) {
  plot(b,select=i,scale=0,scheme=1)
  ii <- b$smooth[[i]]$first.para:b$smooth[[i]]$last.para
  ff <- X[,ii]%*t(br[,ii]) ## posterior curve sample
  fq <- apply(ff,1,quantile,probs=c(.025,.16,.84,.975))
  lines(x,fq[1,],col=2,lty=2);lines(x,fq[4,],col=2,lty=2)
  lines(x,fq[2,],col=2);lines(x,fq[3,],col=2)
}

#####
## Second example, where Gaussian approximation is a failure...
#####

y <- c(rep(0, 89), 1, 0, 1, 0, 0, 1, rep(0, 13), 1, 0, 0, 1,
      rep(0, 10), 1, 0, 0, 1, 1, 0, 1, rep(0,4), 1, rep(0,3),
      1, rep(0, 3), 1, rep(0, 10), 1, rep(0, 4), 1, 0, 1, 0, 0,
      rep(1, 4), 0, rep(1, 5), rep(0, 4), 1, 1, rep(0, 46))
set.seed(3);x <- sort(c(0:10*5,rnorm(length(y)-11)*20+100))
b <- gam(y ~ s(x, k = 15),method = 'REML', family = binomial)
br <- gam.mh(b,thin=2,ns=2000,rw.scale=.4)$bs
X <- model.matrix(b)
par(mfrow=c(1,1))
plot(x, y, col = rgb(0,0,0,0.25), ylim = c(0,1))
ff <- X%*t(br) ## posterior curve sample
linv <- b$family$linkinv
## Get intervals for the curve on the response scale...
fq <- linv(apply(ff,1,quantile,probs=c(.025,.16,.5,.84,.975)))
lines(x,fq[1,],col=2,lty=2);lines(x,fq[5,],col=2,lty=2)
lines(x,fq[2,],col=2);lines(x,fq[4,],col=2)
lines(x,fq[3,],col=4)
## Compare to the Gaussian posterior approximation
fv <- predict(b,se=TRUE)
lines(x,linv(fv$fit))
lines(x,linv(fv$fit-2*fv$se.fit),lty=3)
lines(x,linv(fv$fit+2*fv$se.fit),lty=3)
## ... Notice the useless 95% CI (black dotted) based on the
## Gaussian approximation!

```

gam.models

$y_i \mu_i$

$$g(\mu_i) = \mathbf{X}_i \beta + f_1(x_{1i}, x_{2i}) + f_2(x_{3i}) + L_i f_3(x_4) + \dots$$

$$g\mathbf{X}_i\beta x_jf_jL_if_3$$

```
gamby
gamfamilygamglmgam
```

$$g(\mu_i) = \beta_0 + \beta_1x_{1i} + \beta_2x_{2i} + f_1(x_{3i}) + f_2(x_{4i},x_{5i})$$

```
y_i\mu_i g
gam
y ~ x1 + x2 + s(x3) + s(x4,x5)
stechoose.k
y ~ x1 + x2 + s(x3,bs="cr",k=20) + s(x4,x5,k=26,fx=TRUE)
x_4x_5te
y ~ x1 + x2 + s(x3) + te(x4,x5)
x_4x_5
y ~ x1 + x2 + s(x3) + te(x4,x5,bs=c("cr","ps"),k=c(6,7))
```

$$E(y_i) = f_1(x_i) + f_2(z_i) + f_3(x_i,z_i)$$

$$E(y_i) = f_1(x_i) + f_2(z_i) + f_3(v_i) + f_4(x_i,z_i) + f_5(z_i,v_i) + f_6(z_i,v_i) + f_7(x_i,z_i,v_i)$$

```
ti
y ~ ti(x) + ti(z) + ti(x,z)
y ~ ti(x) + ti(z) + ti(v) + ti(x,z) + ti(x,v) + ti(z,v)+ti(x,z,v)
titis
gamtesti
```

```
by
stebyby_i^{th}_i^{th}
factor.smooth.interactionbyfactororderedidsteorderedbyb3by
```

$$E(y_i) = \beta_0 + f(x_i)z_i$$

```
fz_i
y ~ s(x,by=z)
byz
by
by
bybybyLL%%rep(1,ncol(L))"always.apply"
```

idsteididid

$E(y_i) \equiv \mu_i$

$$g(\mu_i) = f_1(x_{1i}) + f_2(x_{2i}, x_{3i}) + f_3(x_{4i})$$

$f_1 f_3 x_2 x_3$ gam

y ~ s(x1,id=1) + te(x_2,x3) + s(x4,id=1)

ididby

idgamm

mgcvby

s(X,Z,by=L)

$XZL n \times p f_i^{\text{th}}$

$$\sum_{j=1}^p L_{ij} f(X_{ij}, Z_{ij})$$

$LF n \times p XZ$ rowSums(L*F)

teslinear.functional.terms

gams(...,bs="re")smooth.construct.re.smooth.specparaPengamgam.vcomp
random.effectssmooth.construct.re.smooth.specgamm

bygam

$\mathbf{X} \beta \beta^T \mathbf{S}_1 \beta \beta^T \mathbf{S}_2 \beta$

gam(y ~ X - 1,paraPen=list(X=list(S1,S2)))

paraPenparaPenLranksprankLsp

paraPen

freq=TRUEanova.gamsummary.gam

<simon.wood@r-project.org>

require(mgcv)

set.seed(10)

simulate data from $y = f(x_2) * x_1 + \text{error}$

dat <- gamSim(3,n=400)

b<-gam(y ~ s(x2,by=x1),data=dat)

plot(b,pages=1)

summary(b)


```

## Factor `by' variable example (with a spurious covariate x0)
## simulate data...

dat <- gamSim(4)

## fit model...
b <- gam(y ~ fac+s(x2,by=fac)+s(x0),data=dat)
plot(b,pages=1)
summary(b)

## note that the preceding fit is the same as...
b1<-gam(y ~ s(x2,by=as.numeric(fac==1))+s(x2,by=as.numeric(fac==2))+
        s(x2,by=as.numeric(fac==3))+s(x0)-1,data=dat)
## ... the `-1' is because the intercept is confounded with the
## *uncentred* smooths here.
plot(b1,pages=1)
summary(b1)

## repeat forcing all s(x2) terms to have the same smoothing param
## (not a very good idea for these data!)
b2 <- gam(y ~ fac+s(x2,by=fac,id=1)+s(x0),data=dat)
plot(b2,pages=1)
summary(b2)

## now repeat with a single reference level smooth, and
## two `difference' smooths...
dat$fac <- ordered(dat$fac)
b3 <- gam(y ~ fac+s(x2)+s(x2,by=fac)+s(x0),data=dat,method="REML")
plot(b3,pages=1)
summary(b3)

rm(dat)

## An example of a simple random effects term implemented via
## penalization of the parametric part of the model...

dat <- gamSim(1,n=400,scale=2) ## simulate 4 term additive truth
## Now add some random effects to the simulation. Response is
## grouped into one of 20 groups by `fac' and each groups has a
## random effect added...
fac <- as.factor(sample(1:20,400,replace=TRUE))
dat$X <- model.matrix(~fac-1)
b <- rnorm(20)*.5
dat$y <- dat$y + dat$X%*%b

## now fit appropriate random effect model...
PP <- list(X=list(rank=20,diag(20)))
rm <- gam(y~ X+s(x0)+s(x1)+s(x2)+s(x3),data=dat,paraPen=PP)
plot(rm,pages=1)
## Get estimated random effects standard deviation...
sig.b <- sqrt(rm$sig2/rm$sp[1]);sig.b

## a much simpler approach uses "re" terms...

rm1 <- gam(y ~ s(fac,bs="re")+s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="ML")

```

```

gam.vcomp(rm1)

## Simple comparison with lme, using Rail data.
## See ?random.effects for a simpler method
require(nlme)
b0 <- lme(travel~1,data=Rail,~1|Rail,method="ML")
Z <- model.matrix(~Rail-1,data=Rail,
  contrasts.arg=list(Rail="contr.treatment"))
b <- gam(travel~Z,data=Rail,paraPen=list(Z=list(diag(6))),method="ML")

b0
(b$reml.scale/b$sp)^.5 ## `gam' ML estimate of Rail sd
b$reml.scale^.5        ## `gam' ML estimate of residual sd

b0 <- lme(travel~1,data=Rail,~1|Rail,method="REML")
Z <- model.matrix(~Rail-1,data=Rail,
  contrasts.arg=list(Rail="contr.treatment"))
b <- gam(travel~Z,data=Rail,paraPen=list(Z=list(diag(6))),method="REML")

b0
(b$reml.scale/b$sp)^.5 ## `gam' REML estimate of Rail sd
b$reml.scale^.5        ## `gam' REML estimate of residual sd

#####
## Approximate large dataset logistic regression for rare events
## based on subsampling the zeroes, and adding an offset to
## approximately allow for this.
## Doing the same thing, but upweighting the sampled zeroes
## leads to problems with smoothness selection, and CIs.
#####
n <- 50000 ## simulate n data
dat <- gamSim(1,n=n,dist="binary",scale=.33)
p <- binomial()$linkinv(dat$f-6) ## make 1's rare
dat$y <- rbinom(p,1,p)          ## re-simulate rare response

## Now sample all the 1's but only proportion S of the 0's
S <- 0.02                      ## sampling fraction of zeroes
dat <- dat[dat$y==1 | runif(n) < S,] ## sampling

## Create offset based on total sampling fraction
dat$s <- rep(log(nrow(dat)/n),nrow(dat))

lr.fit <- gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+s(x3,bs="cr")+
  offset(s),family=binomial,data=dat,method="REML")

## plot model components with truth overlaid in red
op <- par(mfrow=c(2,2))
fn <- c("f0","f1","f2","f3");xn <- c("x0","x1","x2","x3")
for (k in 1:4) {
  plot(lr.fit,select=k,scale=0)
  ff <- dat[[fn[k]];xx <- dat[[xn[k]]]
  ind <- sort.int(xx,index.return=TRUE)$ix
  lines(xx[ind],(ff-mean(ff))[ind]*.33,col=2)
}
par(op)
rm(dat)

```

```
## A Gamma example, by modify `gamSim' output...

dat <- gamSim(1,n=400,dist="normal",scale=1)
dat$f <- dat$f/4 ## true linear predictor
Ey <- exp(dat$f);scale <- .5 ## mean and GLM scale parameter
## Note that `shape' and `scale' in `rgamma' are almost
## opposite terminology to that used with GLM/GAM...
dat$y <- rgamma(Ey*0,shape=1/scale,scale=Ey*scale)
bg <- gam(y~ s(x0)+ s(x1)+s(x2)+s(x3),family=Gamma(link=log),
          data=dat,method="REML")
plot(bg,pages=1,scheme=1)
```

gam.outer

```
newtonbfgsoptimlmgam.fit3
```

```
gam
```

```
gam.outer(lsp,fscale,family,control,method,optimizer,
          criterion,scale,gamma,G,start=NULL,nei=NULL,...)
```

```
lsp
fscale
family
control      gam.fit3
method        gam
optimizer     gam
criterion     "UBRE""GCV""GACV""REML""P-REML"
scale
gamma
G             mgcv::gam.setup
start
nei           gam
...           gam.fit3
```

```
<simon.wood@r-project.org>
```

```
gam.fit3gammagic
```

`gam.reparam`

```
gam.reparam(rS, lsp, deriv)
```

```
rS          fixed.penalty==TRUElength(rS)>length(sp)rS[[i]]
            totalPenaltySpacemini.roots
lsp
deriv       deriv==1deriv>1
```

```
S
rS
QsS = t(Qs)%*%S0*%QsS0sprS
det
det1deriv >0
det2deriv>1
```

`gam.scale`

```
gamfamilyscale.estgam.control"fletcher""pearson""deviance"
```

```
gamm1mebam"fREML"
```

```
<simon.wood@r-project.org>
```

```
gam.control
```

`gam.selection`

`gam()`method[gam](#)
[gam](#)methodoptimizersp[gam](#)spste

`cs.smooth`tps.smooth"cs""ts"
select[gam](#)

[AIC](#)thetatheta
summary.gam

<simon.wood@r-project.org>

[gamstep.gam](#)

```
## an example of automatic model selection via null space penalization
library(mgcv)
set.seed(3); n<-200
dat <- gamSim(1,n=n,scale=.15,dist="poisson") ## simulate data
dat$x4 <- runif(n, 0, 1); dat$x5 <- runif(n, 0, 1) ## spurious

b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3)+s(x4)+s(x5),data=dat,
      family=poisson,select=TRUE,method="REML")
summary(b)
plot(b,pages=1)
```

gam.side

mgcv:::gam.setup

gam.side(sm,Xp,tol=.Machine\$double.eps^.5,with.pen=TRUE)

sm [smooth.construct](#)

Xp

tol

with.pen

$y \sim s(x) + s(z) + s(x, z)$ [gam](#)

[fixDependence](#)

Xp

"del.index"

$y \sim s(x) + s(z) + ti(x, z)$ $y \sim ti(x) + ti(z) + ti(x, z)$ $y \sim s(x) + s(z) + s(x, z)$

<simon.wood@r-project.org>

[tigam.models](#)

```

## The first two examples here illustrate models that cause
## gam.side to impose constraints, but both are a bad way
## of estimating such models. The 3rd example is the right
## way...
set.seed(7)
require(mgcv)
dat <- gamSim(n=400,scale=2) ## simulate data
## estimate model with redundant smooth interaction (bad idea).
b<-gam(y~s(x0)+s(x1)+s(x0,x1)+s(x2),data=dat)
plot(b,pages=1)

## Simulate data with real interaction...
dat <- gamSim(2,n=500,scale=.1)
old.par<-par(mfrow=c(2,2))

## a fully nested tensor product example (bad idea)
b <- gam(y~s(x,bs="cr",k=6)+s(z,bs="cr",k=6)+te(x,z,k=6),
         data=dat$data)
plot(b)

old.par<-par(mfrow=c(2,2))
## A fully nested tensor product example, done properly,
## so that gam.side is not needed to ensure identifiability.
## ti terms are designed to produce interaction smooths
## suitable for adding to main effects (we could also have
## used s(x) and s(z) without a problem, but not s(z,x)
## or te(z,x)).
b <- gam(y ~ ti(x,k=6) + ti(z,k=6) + ti(x,z,k=6),
         data=dat$data)
plot(b)

par(old.par)
rm(dat)

```

gam.vcomp

```
gam.vcomp(x,rescale=TRUE,conf.lev=.95)
```

```
## S3 method for class 'gam.vcomp'
print(x,...)
```

```

x                gamgam()gam.vcomp
rescale          TRUE
conf.lev
...

```

gam

gam

vcvcvc

all

rankrank.hess

<simon.wood@r-project.org>

smooth.construct.re.smooth.spec

```
set.seed(3)
require(mgcv)
## simulate some data, consisting of a smooth truth + random effects

dat <- gamSim(1,n=400,dist="normal",scale=2)
a <- factor(sample(1:10,400,replace=TRUE))
b <- factor(sample(1:7,400,replace=TRUE))
Xa <- model.matrix(~a-1) ## random main effects
Xb <- model.matrix(~b-1)
Xab <- model.matrix(~a:b-1) ## random interaction
dat$y <- dat$y + Xa%*%rnorm(10)*.5 +
           Xb%*%rnorm(7)*.3 + Xab%*%rnorm(70)*.7
dat$a <- a;dat$b <- b

## Fit the model using "re" terms, and smoother linkage

mod <- gam(y~s(a,bs="re")+s(b,bs="re")+s(a,b,bs="re")+s(x0,id=1)+s(x1,id=1)+
           s(x2,k=15)+s(x3),data=dat,method="ML")

gam.vcomp(mod)
```

gam2objective

[optimnlmgam.outer](#)

gam2objective(lsp,args,...)
gam2derivative(lsp,args,...)

lsp

args [gam.fit3](#)

... [gam.fit3](#)

gam2objectivegam2derivative[optim](#)

gam4objectivegam2objectivenlm

<simon.wood@r-project.org>

[gam.fit3gammagic](#)

gamlss.etamu

g(mu) = lplpgmulp[trind.generator](#)

gamlss.etamu(l1, l2, l3 = NULL, l4 = NULL, ig1, g2, g3 = NULL,
g4 = NULL, i2, i3 = NULL, i4 = NULL, deriv = 0)

```

l1
l2
l3
l4
ig1
g2
g3
g4
i2      l2[,i2[i,j]]
i3      l3[,i3[i,j,k]]
i4      l4[,i4[i,j,k,l]]
deriv   deriv==0deriv==1deriv==2

l1l2l3l4

```

[trind.generator](#)

gamlss.gH

[trind.generator](#)

```

gamlss.gH(X, jj, l1, l2, i2, l3 = 0, i3 = 0, l4 = 0, i4 = 0, d1b = 0,
  d2b = 0, deriv = 0, fh = NULL, D = NULL,sandwich=FALSE)

```

```

X
jj      X[,jj[[i]]]
l1
l2
i2      l2[,i2[i,j]]
l3
i3      l3[,i3[i,j,k]]
l4
i4      l4[,i4[i,j,k,l]]
d1b
d2b
deriv   deriv==0deriv==1deriv==2deriv==3
fh
D
sandwich TRUEl2

```

lb1bbd1HtrHid2H

[trind.generator](#)

gamm

[lme](#)[gamm](#)[PQL](#)[glm](#)[PQL](#)[MASS](#)[gam](#)

[lme4](#)[nlme](#)[gamm4](#)[gamm4](#)

[gam](#)[lme](#)

[gam](#)[lme](#)

```
gamm(formula, random=NULL, correlation=NULL, family=gaussian(),
data=list(), weights=NULL, subset=NULL, na.action, knots=NULL,
control=list(niterEM=0, optimMethod="L-BFGS-B", returnObject=TRUE),
niterPQL=20, verbosePQL=TRUE, method="ML", drop.unused.levels=TRUE,
mustart=NULL, etastart=NULL,...)
```

formula [formula.gam](#)[gam.models](#)[glm](#)[steid](#)

random [lme](#)[list](#)[gamm](#)

correlation [corStruct](#)[corClasses](#)[lme](#)[lme](#)

family family[glm](#)[gam](#)[gaussian](#)[gamm](#)[lme](#)[gamm](#)[PQL](#)

data environment(formula)[gamm](#)

weights [glm](#)[lme](#)[lme](#)

subset

na.action

knots

control [lme](#)[lmeControl](#)[lme](#)[pdMat](#)[niterEM=0](#)[optimMethod](#)[nlminb](#)

niterPQL

verbosePQL

method "ML" "REML" [lme](#)[gamm](#)[PQL](#)

drop.unused.levels

mustart

etastart mustart

... [lme](#)

lme^{glmmPQLnlme}
lme^{gammpQLgammgamm}
^{gam}gam^{predict.gam}
lmeoptions(mgcv.vc.logrange)^{notExp2}niterEMcontrolgammnlme

gam gam^{predictsummaryprintvis.gamanovagammpQL}
lme lme^{gammpQLlme}gammpQL

gamm^{gam}gam^{gammagamm}
gamm^{gams(...,bs="re")}gamm4
gamm^{glmmPQLMASSlme}

random
gamm^{gamlme}
gamm^{gammemory.limit}

gam^{gamObject}glmanova
gamm^{notExp2}

<simon.wood@r-project.org>

[magictespredict.gamplot.gamsummary.gamnegbinvis.gampdTensgamm4https://cran.r-project.org/package=gamm4](https://cran.r-project.org/package=gamm4)

```
library(mgcv)
## simple examples using gamm as alternative to gam
set.seed(0)
dat <- gamSim(1,n=200,scale=2)
b <- gamm(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b$gam,pages=1)
summary(b$lme) # details of underlying lme fit
summary(b$gam) # gam style summary of fitted model
anova(b$gam)
gam.check(b$gam) # simple checking plots

b <- gamm(y~te(x0,x1)+s(x2)+s(x3),data=dat)
op <- par(mfrow=c(2,2))
plot(b$gam)
par(op)
rm(dat)

## Add a factor to the linear predictor, to be modelled as random
dat <- gamSim(6,n=200,scale=.2,dist="poisson")
b2 <- gamm(y~s(x0)+s(x1)+s(x2),family=poisson,
           data=dat,random=list(fac=~1))
plot(b2$gam,pages=1)
fac <- dat$fac
rm(dat)
vis.gam(b2$gam)

## In the generalized case the 'gam' object is based on the working
## model used in the PQL fitting. Residuals for this are not
## that useful on their own as the following illustrates...

gam.check(b2$gam)

## But more useful residuals are easy to produce on a model
## by model basis. For example...

fv <- exp(fitted(b2$lme)) ## predicted values (including re)
rsd <- (b2$gam$y - fv)/sqrt(fv) ## Pearson residuals (Poisson case)
op <- par(mfrow=c(1,2))
qqnorm(rsd);plot(fv^.5,rsd)
par(op)

## now an example with autocorrelated errors....
n <- 200;sig <- 2
x <- 0:(n-1)/(n-1)
f <- 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
e <- rnorm(n,0,sig)
for (i in 2:n) e[i] <- 0.6*e[i-1] + e[i]
y <- f + e
op <- par(mfrow=c(2,2))
## Fit model with AR1 residuals
```

```

b <- gamm(y~s(x,k=20),correlation=corAR1())
plot(b$gam);lines(x,f-mean(f),col=2)
## Raw residuals still show correlation, of course...
acf(residuals(b$gam),main="raw residual ACF")
## But standardized are now fine...
acf(residuals(b$lme,type="normalized"),main="standardized residual ACF")
## compare with model without AR component...
b <- gam(y~s(x,k=20))
plot(b);lines(x,f-mean(f),col=2)

## more complicated autocorrelation example - AR errors
## only within groups defined by `fac`
e <- rnorm(n,0,sig)
for (i in 2:n) e[i] <- 0.6*e[i-1]*(fac[i-1]==fac[i]) + e[i]
y <- f + e
b <- gamm(y~s(x,k=20),correlation=corAR1(form=~1|fac))
plot(b$gam);lines(x,f-mean(f),col=2)
par(op)

## more complex situation with nested random effects and within
## group correlation

set.seed(0)
n.g <- 10
n<-n.g*10*4
## simulate smooth part...
dat <- gamSim(1,n=n,scale=2)
f <- dat$f
## simulate nested random effects....
fa <- as.factor(rep(1:10,rep(4*n.g,10)))
ra <- rep(rnorm(10),rep(4*n.g,10))
fb <- as.factor(rep(rep(1:4,rep(n.g,4)),10))
rb <- rep(rnorm(4),rep(n.g,4))
for (i in 1:9) rb <- c(rb,rep(rnorm(4),rep(n.g,4)))
## simulate auto-correlated errors within groups
e<-array(0,0)
for (i in 1:40) {
  eg <- rnorm(n.g, 0, sig)
  for (j in 2:n.g) eg[j] <- eg[j-1]*0.6+ eg[j]
  e<-c(e,eg)
}
dat$y <- f + ra + rb + e
dat$fa <- fa;dat$fb <- fb
## fit model ....
b <- gamm(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+
  s(x3,bs="cr"),data=dat,random=list(fa=~1,fb=~1),
  correlation=corAR1())
plot(b$gam,pages=1)
summary(b$gam)
vis.gam(b$gam)

## Prediction from gam object, optionally adding
## in random effects.

## Extract random effects and make names more convenient...
refa <- ranef(b$lme,level=5)
rownames(refa) <- substr(rownames(refa),start=9,stop=20)

```

```

refb <- ranef(b$lme, level=6)
rownames(refb) <- substr(rownames(refb), start=9, stop=20)

## make a prediction, with random effects zero...
p0 <- predict(b$gam, data.frame(x0=.3, x1=.6, x2=.98, x3=.77))

## add in effect for fa = "2" and fb="2/4"...
p <- p0 + refa["2", 1] + refb["2/4", 1]

## and a "spatial" example...
library(nlme); set.seed(1); n <- 100
dat <- gamSim(2, n=n, scale=0) ## standard example
attach(dat)
old.par <- par(mfrow=c(2, 2))
contour(truth$x, truth$z, truth$f) ## true function
f <- data$f ## true expected response
## Now simulate correlated errors...
cstr <- corGaus(.1, form = ~x+z)
cstr <- Initialize(cstr, data.frame(x=data$x, z=data$z))
V <- corMatrix(cstr) ## correlation matrix for data
Cv <- chol(V)
e <- t(Cv) %*% rnorm(n)*0.05 # correlated errors
## next add correlated simulated errors to expected values
data$y <- f + e ## ... to produce response
b <- gamm(y~s(x, z, k=50), correlation=corGaus(.1, form=~x+z),
          data=data)
plot(b$gam) # gamm fit accounting for correlation
# overfits when correlation ignored....
b1 <- gamm(y~s(x, z, k=50), data=data); plot(b1$gam)
b2 <- gam(y~s(x, z, k=50), data=data); plot(b2)
par(old.par)

```

`gammals`

`gammals $\mu\phi\phi\mu^2$ gam`

`gammals(link=list("identity", "log"), b=-7)`

`link`

`b`

`gamidentitylog $b\eta\phi\log\phi = b + \log(1 + e^{\eta-b})$`

`gam`

`predict.gamtype="link""response" type="response" type="link" type="response" type="link"`

general.family

```
library(mgcv)
## simulate some data
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
      (10 * x)^3 * (1 - x)^10
f3 <- function(x) 0 * x
n <- 400;set.seed(9)
x0 <- runif(n);x1 <- runif(n);
x2 <- runif(n);x3 <- runif(n);
mu <- exp((f0(x0)+f2(x2))/5)
th <- exp(f1(x1)/2-2)
y <- rgamma(n,shape=1/th,scale=mu*th)

b1 <- gam(list(y~s(x0)+s(x2),~s(x1)+s(x3)),family=gammals)
plot(b1,pages=1)
summary(b1)
gam.check(b1)
plot(mu,fitted(b1)[,1]);abline(0,1,col=2)
plot(log(th),fitted(b1)[,2]);abline(0,1,col=2)
```

gamObject

gam"gam""glm""lm"anova logLik influence plot predict print residual s summary
"glm""lm"

gam

aic

assign pterms

boundary

call updategam

cmX

coefficients

control gam

converged

data "glm"gam control keepData TRUE FALSE

db.drho

deviance	
df.null	
df.residual	
edf	
edf1	
edf2	edf1edf2
family	
fitted.values	
formula	
full.sp	min.spgamspspgam
F	
gcv.ubre	
hat	
iter	
linear.predictors	
method	"GCV""UBRE""REML""P-REML""ML""P-ML""PQL""lme.ML""lme.REML"
mgcv.conv	"magic""outer"full.rankrankfully.convergedTRUEFALSEhess.pos.def iterscore.callsrms.grad
min.edf	
model	
na.action	na.action
nsdf	
null.deviance	
offset	
optimizer	optimizergam"magic"
outer.info	gamoptimizernlmoptim
paraPen	paraPengamNULL
pred.formula	predict.gam
prior.weights	
pterm	terms
R	
rank	
reml.scale	
residuals	
rV	rV%%t(rV)*sig2
scale	sig2
scale.estimated	TRUEFALSE
sig2	
smooth	smooth.construct

sp	full.sp
terms	termsmodel
var.summary	vis.gam
Ve	
Vp	
Vc	Vp
weights	
y	

[<simon.wood@r-project.org>](mailto:simon.wood@r-project.org)

[gam](#)

gamSim

[gangamm](#)

```
gamSim(eg=1,n=400,dist="normal",scale=2,verbose=TRUE)
```

```
eg
n
dist
scale
verbose
```

eg

<simon.wood@r-project.org>

[gam](#)[gamm](#)

see ?gam

[gaulss](#)

[gaulss](#)[gam](#)

`gaulss(link=list("identity","logb"),b=0.01)`

link

b "logb"

[gam](#)[gam](#)

"identity""inverse""log""sqrt""logb" $\eta = \log(\sigma - b)\sigma = b + \exp(\eta)\tau = \sigma^{-1}$

[predict.gam](#)`type="link""response" type="response" type="link" $\log(\sigma - b)$` [plot.gam](#) σ

general.family

```
library(mgcv);library(MASS)
b <- gam(list(accel~s(times,k=20,bs="ad"),~s(times)),
          data=mcycle,family=gaulss())
summary(b)
plot(b,pages=1,scale=0)
```

get.var

NULL

```
get.var(txt,data,vecMat=TRUE)
```

```
txt          datadataNULL
```

```
data
```

```
vecMat
```

NULL

<simon.wood@r-project.org>

gam

```
require(mgcv)
y <- 1:4;dat<-data.frame(x=5:10)
get.var("x",dat)
get.var("y",dat)
get.var("x==6",dat)
dat <- list(X=matrix(1:6,3,2))
get.var("X",dat)
```

gevlss

gevlssgam

gevlss(link=list("identity","identity","logit"))

link

gam

$$t(y)^{\xi+1}e^{-t(y)}/\sigma$$

$$t(y) = \{1 + \xi(y - \mu)/\sigma\}^{-1/\xi} \xi \neq 0, t(y) = \exp\{-(y - \mu)/\sigma\} \xi = 0$$

$$\text{gam} \mu \rho = \log(\sigma) \xi$$

$$\text{"identity"} \text{"log"} \mu \rho = \log(\sigma) \xi$$

$$\xi = 0 \xi$$

general.family

```
library(mgcv)
Fi.gev <- function(z,mu,sigma,xi) {
  ## GEV inverse cdf.
  xi[abs(xi)<1e-8] <- 1e-8 ## approximate xi=0, by small xi
  x <- mu + ((-log(z))^-xi-1)*sigma/xi
}

## simulate test data...
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10
set.seed(1)
n <- 500
x0 <- runif(n);x1 <- runif(n);x2 <- runif(n)
mu <- f2(x2)
```

```

rho <- f0(x0)
xi <- (f1(x1)-4)/9
y <- Fi.gev(runif(n),mu,exp(rho),xi)
dat <- data.frame(y,x0,x1,x2);pairs(dat)

## fit model....
b <- gam(list(y~s(x2),~s(x0),~s(x1)),family=gevlss,data=dat)

## same fit using the extended Fellner-Schall method which
## can provide improved numerical robustness...
b <- gam(list(y~s(x2),~s(x0),~s(x1)),family=gevlss,data=dat,
           optimizer="efs")

## plot and look at residuals...
plot(b,pages=1,scale=0)
summary(b)

par(mfrow=c(2,2))
mu <- fitted(b)[,1];rho <- fitted(b)[,2]
xi <- fitted(b)[,3]
## Get the predicted expected response...
fv <- mu + exp(rho)*(gamma(1-xi)-1)/xi
rsd <- residuals(b)
plot(fv,rsd);qqnorm(rsd)
plot(fv,residuals(b,"pearson"))
plot(fv,residuals(b,"response"))

```

gfam

[gambam](#)gfam

by[gam.models](#)

gfam(f1)

f1 familyextended.familygam

gfamf1

gfamextended.family

extended.familyextended.familyf1

[qq.gam](#)

[ocat](#)"response"gfam

gfam

extended.family

<simon.wood@r-project.org>

```
library(mgcv)
## a mixed family simulator function to play with...
sim.gfam <- function(dist,n=400) {
  ## dist can be norm, pois, gamma, binom, nbinom, tw, ocat (R assumed 4)
  ## links used are identity, log or logit.
  dat <- gamSim(1,n=n,verbose=FALSE)
  nf <- length(dist) ## how many families
  fin <- c(1:nf,sample(1:nf,n-nf,replace=TRUE)) ## family index
  dat[,6:10] <- dat[,6:10]/5 ## a scale that works for all links used
  y <- dat$y;
  for (i in 1:nf) {
    ii <- which(fin==i) ## index of current family
    ni <- length(ii); fi <- dat$fi[ii]
    if (dist[i]=="norm") {
      y[ii] <- fi + rnorm(ni)*.5
    } else if (dist[i]=="pois") {
      y[ii] <- rpois(ni,exp(fi))
    } else if (dist[i]=="gamma") {
      scale <- .5
      y[ii] <- rgamma(ni,shape=1/scale,scale=exp(fi)*scale)
    } else if (dist[i]=="binom") {
      y[ii] <- rbinom(ni,1,binomial()$linkinv(fi))
    } else if (dist[i]=="nbinom") {
      y[ii] <- rnbinom(ni,size=3,mu=exp(fi))
    } else if (dist[i]=="tw") {
      y[ii] <- rTweedie(exp(fi),p=1.5,phi=1.5)
    } else if (dist[i]=="ocat") {
      alpha <- c(-Inf,1,2,2.5,Inf)
      R <- length(alpha)-1
      yi <- fi
      u <- runif(ni)
      u <- yi + log(u/(1-u))
      for (j in 1:R) {
        yi[u > alpha[j]&u <= alpha[j+1]] <- j
      }
      y[ii] <- yi
    }
  }
  dat$y <- cbind(y,fin)
  dat
} ## sim.gfam

## some examples

dat <- sim.gfam(c("binom","tw","norm"))
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),
        family=gfam(list(binomial,tw,gaussian)),data=dat)
predict(b,data.frame(y=1:3,x0=c(.5,.5,.5),x1=c(.3,.2,.3),
                     x2=c(.2,.5,.8),x3=c(.1,.5,.9)),type="response",se=TRUE)
```

```
summary(b)
plot(b,pages=1)

## set up model so that only the binomial observations depend
## on x0...

dat$id1 <- as.numeric(dat$y[,2]==1)
b1 <- gam(y~s(x0,by=id1)+s(x1)+s(x2)+s(x3),
          family=gfam(list(binomial,tw,gaussian)),data=dat)
plot(b1,pages=1) ## note the CI width increase
```

ginla

[gambam](#)

```
ginla(G,A=NULL,nk=16,nb=100,J=1,interactive=FALSE,integ=0,approx=0)
```

```
G          gam(...,fit=FALSE)bam(...,discrete=TRUE,fit=FALSE)
A          NULL
nk
nb
J
interactive >0TRUE2
integ
approx
```

$$\beta\theta y\pi(\beta_i|\theta,y)\pi(\theta|y)\pi(\beta_i|y)\pi(\beta_i|\theta,y)\pi(\theta|y)\theta\pi(\beta_i|\theta,y)\beta_i\beta^*|\beta_i\log\pi(\beta,\theta,y)\beta\pi(\beta|y)\beta_i\textcolor{blue}{gam}\beta\beta_i$$

$$H[-i,-i]H\textcolor{blue}{choldrop}$$

```
approxapprox==1approx==2
interactive
bamdisrete=TRUEdiscrete=FALSE
```

```
betadensitynbint!=0rem1
```

ginla

<simon.wood@r-project.org>


```

require(mgcv); require(MASS)

## example using a scale location model for the motorcycle data. A simple
## plotting routine is produced first...

plot.inla <- function(x,inla,k=1,levels=c(.025,.1,.5,.9,.975),
                     lcol = c(2,4,4,4,2),lwd = c(1,1,2,1,1),lty=c(1,1,1,1,1),
                     xlab="x",ylab="y",cex.lab=1.5) {
  ## a simple effect plotter, when distributions of function values of
  ## 1D smooths have been computed
  require(splines)
  p <- length(x)
  betaq <- matrix(0,length(levels),p) ## storage for beta quantiles
  for (i in 1:p) { ## work through x and betas
    j <- i + k - 1
    p <- cumsum(inla$density[j,])*(inla$beta[j,2]-inla$beta[j,1])
    ## getting quantiles of function values...
    betaq[i,i] <- approx(p,y=inla$beta[j,],levels)$y
  }
  xg <- seq(min(x),max(x),length=200)
  ylim <- range(betaq)
  ylim <- 1.1*(ylim-mean(ylim))+mean(ylim)
  for (j in 1:length(levels)) { ## plot the quantiles
    din <- interpSpline(x,betaq[j,])
    if (j==1) {
      plot(xg,predict(din,xg)$y,ylim=ylim,type="l",col=lcol[j],
            xlab=xlab,ylab=ylab,lwd=lwd[j],cex.lab=1.5,lty=lty[j])
    } else lines(xg,predict(din,xg)$y,col=lcol[j],lwd=lwd[j],lty=lty[j])
  }
} ## plot.inla

## set up the model with a `gam' call...

G <- gam(list(accel~s(times,k=20,bs="ad"),~s(times)),
          data=mcycle,family=gaulss(),fit=FALSE)
b <- gam(G=G,method="REML") ## regular GAM fit for comparison

## Now use ginla to get posteriors of estimated effect values
## at evenly spaced times. Create A matrix for this...

rat <- range(mcycle$times)
pd0 <- data.frame(times=seq(rat[1],rat[2],length=20))
X0 <- predict(b,newdata=pd0,type="lpmatrix")
X0[,21:30] <- 0
pd1 <- data.frame(times=seq(rat[1],rat[2],length=10))
X1 <- predict(b,newdata=pd1,type="lpmatrix")
X1[,1:20] <- 0
A <- rbind(X0,X1) ## A maps coefs to required function values

## call ginla. Set integ to 1 for integrated version.
## Set interactive = 1 or 2 to plot marginal posterior distributions

```

```
## (red) and simple Gaussian approximation (black).

inla <- ginla(G,A,integ=0)

par(mfrow=c(1,2),mar=c(5,5,1,1))
fv <- predict(b,se=TRUE) ## usual Gaussian approximation, for comparison

## plot inla mean smooth effect...
plot.inla(pd0$times,inla,k=1,xlab="time",ylab=expression(f[1](time)))

## overlay simple Gaussian equivalent (in grey) ...
points(mcycle$times,mcycle$accel,col="grey")
lines(mcycle$times,fv$fit[,1],col="grey",lwd=2)
lines(mcycle$times,fv$fit[,1]+2*fv$se.fit[,1],lty=2,col="grey",lwd=2)
lines(mcycle$times,fv$fit[,1]-2*fv$se.fit[,1],lty=2,col="grey",lwd=2)

## same for log sd smooth...
plot.inla(pd1$times,inla,k=21,xlab="time",ylab=expression(f[2](time)))
lines(mcycle$times,fv$fit[,2],col="grey",lwd=2)
lines(mcycle$times,fv$fit[,2]+2*fv$se.fit[,2],col="grey",lty=2,lwd=2)
lines(mcycle$times,fv$fit[,2]-2*fv$se.fit[,2],col="grey",lty=2,lwd=2)

## ... notice some real differences for the log sd smooth, especially
## at the lower and upper ends of the time interval.
```

gumb1s

gumb1sgam

gumb1s(link=list("identity","log"),b=-7)

link $\mu\beta$
b

$$z = (y - \mu)e^{-\beta l} = -\beta - z - e^{-z}\mu + \gamma e^{\beta}\gamma\pi^2 e^{2\beta}/6$$

$$\text{gumb1sgam}\mu\beta\text{identity}\beta\log b\eta\beta\beta = b + \log(1 + e^{\eta-b})$$

gam $\mu\beta$

β predict.gamtype"link""response"type="response"type="link"type="response"
type="link"

general.family

```

library(mgcv)
## simulate some data
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
      (10 * x)^3 * (1 - x)^10
n <- 400; set.seed(9)
x0 <- runif(n); x1 <- runif(n);
x2 <- runif(n); x3 <- runif(n);
mu <- f0(x0) + f1(x1)
beta <- exp(f2(x2)/5)
y <- mu - beta * log(-log(runif(n))) ## Gumbel quantile function

b <- gam(list(y~s(x0)+s(x1), ~s(x2)+s(x3)), family=gumbels)
plot(b, pages=1, scale=0)
summary(b)
gam.check(b)

```

identifiability

byby

$$\sum_i f(x_i) = 0.$$

byby

[pcstetit2](#)

```

## Example of three groups, each with a different smooth dependence on x
## but each starting at the same value...
require(mgcv)
set.seed(53)
n <- 100; x <- runif(3*n); z <- runif(3*n)
fac <- factor(rep(c("a", "b", "c"), each=100))
y <- c(sin(x[1:100]*4), exp(3*x[101:200])/10-.1, exp(-10*(x[201:300]-.5))/
      (1+exp(-10*(x[201:300]-.5)))-0.9933071) + z*(1-z)*5 + rnorm(100)*.4

## 'pc' used to constrain smooths to 0 at x=0...
b <- gam(y~s(x, by=fac, pc=0)+s(z))
plot(b, pages=1)

```

in.out

in.out(bnd,x)

bnd NAbnd[[i]][[1]]bnd[[i]][[2]]
x bnd

nrow(x)TRUExFALSE

<simon.wood@r-project.org>

```
library(mgcv)
data(columb.polys)
bnd <- columb.polys[[2]]
plot(bnd,type="n")
polygon(bnd)
x <- seq(7.9,8.7,length=20)
y <- seq(13.7,14.3,length=20)
gr <- as.matrix(expand.grid(x,y))
inside <- in.out(bnd,gr)
points(gr,col=as.numeric(inside)+1)
```

influence.gam

gam

```
## S3 method for class 'gam'
influence(model,...)
```

model gamgam()
...

hat

<simon.wood@r-project.org>

gam

initial.sp

initial.sp(X,S,off,expensive=FALSE,XX=FALSE)

X	
S	$S[[i]]S[[i]]\text{off}[i]\text{off}[i]S[[i]]$
off	$S[[i]]$
expensive	TRUE
XX	$\text{TRUE}X^TXX$

magicgam

<simon.wood@r-project.org>

magicgam.outergam

inSide

```
inSide(bnd,x,y,xname=NULL,yname=NULL)
```

```
bnd          xyNAbnd
x
y
xname        bndxx
yname        bndyy
```

```
NA
```

```
x, y
```

```
xyTRUEx, y
```

```
<simon.wood@r-project.org>
```

```
require(mgcv)
m <- 300;n <- 150
xm <- seq(-1,4,length=m);yn <- seq(-1,1,length=n)
x <- rep(xm,n);y <- rep(yn,rep(m,n))
er <- matrix(fs.test(x,y),m,n)
bnd <- fs.boundary()
in.bnd <- inSide(bnd,x,y)
plot(x,y,col=as.numeric(in.bnd)+1,pch=".")
lines(bnd$x,bnd$y,col=3)
points(x,y,col=as.numeric(in.bnd)+1,pch=".")
## check boundary details ...
plot(x,y,col=as.numeric(in.bnd)+1,pch=".",ylim=c(-1,0),xlim=c(3,3.5))
lines(bnd$x,bnd$y,col=3)
points(x,y,col=as.numeric(in.bnd)+1,pch=".")

## alternatively, give the names of x and y
d <- data.frame(x, y); rm(x, y)
in.bnd2 <- inSide(bnd, d$x, d$y, xname="x", yname="y")
all.equal(in.bnd, in.bnd2)
```

`interpret.gam`

`mgcvgam`

`interpret.gam(gf, extra.special = NULL)`

`gf` [gamgammgam](#)
`extra.special` `stetit2`

`split.gam.formula`

`pf`
`pfok` `pf`
`smooth.spec` `xx.smooth.specxx`
`full.formula`
`fake.formula`
`response`

`<simon.wood@r-project.org>`

[gamgamm](#)

`jagam`

`jagam`

`jagam(formula,family=gaussian,data=list(),file,weights=NULL,na.action,
offset=NULL,knots=NULL,sp=NULL,drop.unused.levels=TRUE,
control=gam.control(),centred=TRUE,sp.prior = "gamma",diagonalize=FALSE)`

`sim2jam(sam,pregam,edf.type=2,burnin=0)`

```

formula      formula.gamgam.modelsstetit2
family       glmfamily
data         environment(formula)jagam
file         setwd
weights
na.action
offset       formulalmglm
control      gam.controljagam
knots        kktprs"tp"/"ts"
sp           length(sp)
drop.unused.levels

```

```

centred      FALSEFALSE
sp.prior     "gamma""log.uniform"
diagonalize  "glm"
sam          brhomu
pregam       mgcvjagam
edf.type
burnin

```

```
jagamcentredcentred=FALSE
```

```
mgcv
```

```

sim2jamgampregamjagamrjagsgamsim2gamedf.type=0jagamedf.type=1gamedf.type=2Vpsim
muXWX = t(X)%*%W%*%XrowSums(Vp*XWX)*scalemuXWVpjagam

```

```
jagam
```

```
pregam      mgcv
```

```
jags.data
```

```
jags.ini
```

```
sim2jam"jam"mgcvgamObject
```

```
<simon.wood@r-project.org>
```


gamgambam

```
## the following illustrates a typical workflow. To run the
## 'Not run' code you need rjags (and JAGS) to be installed.
require(mgcv)

set.seed(2) ## simulate some data...
n <- 400
dat <- gamSim(1,n=n,dist="normal",scale=2)
## regular gam fit for comparison...
b0 <- gam(y~s(x0)+s(x1) + s(x2)+s(x3),data=dat,method="REML")

## Set directory and file name for file containing jags code.
## In real use you would *never* use tempdir() for this. It is
## only done here to keep CRAN happy, and avoid any chance of
## an accidental overwrite. Instead you would use
## setwd() to set an appropriate working directory in which
## to write the file, and just set the file name to what you
## want to call it (e.g. "test.jags" here).

jags.file <- paste(tempdir(),"/test.jags",sep="")

## Set up JAGS code and data. In this one might want to diagonalize
## to use conjugate samplers. Usually call 'setwd' first, to set
## directory in which model file ("test.jags") will be written.

jd <- jagam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,file=jags.file,
            sp.prior="gamma",diagonalize=TRUE)

## In normal use the model in "test.jags" would now be edited to add
## the non-standard stochastic elements that require use of JAGS....

## Not run:
require(rjags)
load.module("glm") ## improved samplers for GLMs often worth loading
jm <- jags.model(jags.file,data=jd$jags.data,init=jd$jags.ini,n.chains=1)
list.samplers(jm)
sam <- jags.samples(jm,c("b","rho","scale"),n.iter=10000,thin=10)
jam <- sim2jam(sam,jd$pregam)
plot(jam,pages=1)
jam
pd <- data.frame(x0=c(.5,.6),x1=c(.4,.2),x2=c(.8,.4),x3=c(.1,.1))
fv <- predict(jam,newdata=pd)
## and some minimal checking...
require(coda)
effectiveSize(as.mcmc.list(sam$b))
```

```

## End(Not run)

## a gamma example...
set.seed(1); n <- 400
dat <- gamSim(1,n=n,dist="normal",scale=2)
scale <- .5; Ey <- exp(dat$f/2)
dat$y <- rgamma(n,shape=1/scale,scale=Ey*scale)
jd <- jagam(y~s(x0)+te(x1,x2)+s(x3),data=dat,family=Gamma(link=log),
            file=jags.file,sp.prior="log.uniform")

## In normal use the model in "test.jags" would now be edited to add
## the non-standard stochastic elements that require use of JAGS...

## Not run:
require(rjags)
## following sets random seed, but note that under JAGS 3.4 many
## models are still not fully repeatable (JAGS 4 should fix this)
jd$jags.ini$.RNG.name <- "base::Mersenne-Twister" ## setting RNG
jd$jags.ini$.RNG.seed <- 6 ## how to set RNG seed
jm <- jags.model(jags.file,data=jd$jags.data,init=jd$jags.ini,n.chains=1)
list.samplers(jm)
sam <- jags.samples(jm,c("b","rho","scale","mu"),n.iter=10000,thin=10)
jam <- sim2jam(sam,jd$pregam)
plot(jam,pages=1)
jam
pd <- data.frame(x0=c(.5,.6),x1=c(.4,.2),x2=c(.8,.4),x3=c(.1,.1))
fv <- predict(jam,newdata=pd)

## End(Not run)

```

k.check

gangam()

k.check(b, subsample=5000, n.rep=400)

b gangam()
subsample
n.rep

k-indexp-valuen.repsubsamplesubsamplekedfk\'NA
kedfkchoose.k

<simon.wood@r-project.org>

[choose.kgamgam.check](#)

```
library(mgcv)
set.seed(0)
dat <- gamSim(1,n=200)
b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b,pages=1)
k.check(b)
```

ldetS

S1.setup

```
ldetS(S1, rho, fixed, np, root = FALSE,Stot=FALSE,repara = TRUE,
      nt = 1,deriv=2,sparse=FALSE)
```

S1	S1.setup
rho	
fixed	
np	
root	E
Stot	
repara	gam.reparam
nt	
deriv	
sparse	ES

```
ldetS
ldetS1
ldetS2
S1
rp
Et(E)%*%E = S_totroot==TRUE
SStot==TRUE
```

ldTweedie

$\text{phiprho} = \log(\phi) \theta \exp = (a + b \exp(\theta)) / (1 + \exp(\theta))$

`ldTweedie(y, mu=y, p=1.5, phi=1, rho=NA, theta=NA, a=1.001, b=1.999, all.derivs=FALSE)`

y	
mu	y
p	yppphi=1
phi	yphi*mu^p
rho	phitheta
theta	p = (a+b*exp(theta))/(1+exp(theta))prho
a	ptheta
b	ptheta
all.derivs	TRUEmurhophi

NN

`ldTweedieptweedie`

`rhothetapphiphip=1`

`pphiithetarhomu`

`all.derivs=TRUEyp=1phippip`

`rhothetaall.derivs=TRUEmumuthetamurhomu`

`<simon.wood@r-project.org>`

```
library(mgcv)
## convergence to Poisson illustrated
## notice how p>1.1 is OK
y <- seq(1e-10, 10, length=1000)
p <- c(1.0001, 1.001, 1.01, 1.1, 1.2, 1.5, 1.8, 2)
phi <- .5
fy <- exp(ldTweedie(y, mu=2, p=p[1], phi=phi)[, 1])
plot(y, fy, type="l", ylim=c(0, 3), main="Tweedie density as p changes")
for (i in 2:length(p)) {
  fy <- exp(ldTweedie(y, mu=2, p=p[i], phi=phi)[, 1])
  lines(y, fy, col=i)
}
```

linear.functional.terms

gam

$$g(\mu_i) = \dots + \sum_j L_{ij} f(x_{ij}) + \dots$$

$x_{ij} L_{ij}$

stebyrowSums(F)by L_{ij} rowSums(L*F)

L1rowSums(L)byrowSums(L)

predict.gamnewdatabybyL1

matrixdata.matrix

<simon.wood@r-project.org>

```
### matrix argument 'linear operator' smoothing
library(mgcv)
set.seed(0)

#####
## simple summation example...#
#####

n<-400
sig<-2
x <- runif(n, 0, .9)
f2 <- function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
x1 <- x + .1

f <- f2(x) + f2(x1) ## response is sum of f at two adjacent x values
y <- f + rnorm(n)*sig

X <- matrix(c(x,x1),n,2) ## matrix covariate contains both x values
b <- gam(y~s(X))

plot(b) ## reconstruction of f
plot(f,fitted(b))

## example of prediction with summation convention...
predict(b,list(X=X[1:3,]))

## example of prediction that simply evaluates smooth (no summation)...
predict(b,data.frame(X=c(.2,.3,.7)))

#####
## Simple random effect model example.
## model: y[i] = f(x[i]) + b[k[i]] - b[j[i]] + e[i]
## k[i] and j[i] index levels of i.i.d. random effects, b.
#####
```

```

set.seed(7)
n <- 200
x <- runif(n) ## a continuous covariate

## set up a `factor matrix'...
fac <- factor(sample(letters,n*2,replace=TRUE))
dim(fac) <- c(n,2)

## simulate data from such a model...
nb <- length(levels(fac))
b <- rnorm(nb)
y <- 20*(x-.3)^4 + b[fac[,1]] - b[fac[,2]] + rnorm(n)*.5

L <- matrix(-1,n,2);L[,1] <- 1 ## the differencing 'by' variable

mod <- gam(y ~ s(x) + s(fac,by=L,bs="re"),method="REML")
gam.vcomp(mod)
plot(mod,page=1)

## example of prediction using matrices...
dat <- list(L=L[1:20,],fac=fac[1:20,],x=x[1:20],y=y[1:20])
predict(mod,newdata=dat)

#####
## multivariate integral example. Function `test1' will be integrated#
## (by midpoint quadrature) over 100 equal area sub-squares covering #
## the unit square. Noise is added to the resulting simulated data. #
## `test1' is estimated from the resulting data using two alternative#
## smooths. #
#####

test1 <- function(x,z,sx=0.3,sz=0.4)
{ (pi*sx*sz)*(1.2*exp(-(x-0.2)^2/sx^2-(z-0.3)^2/sz^2)+
  0.8*exp(-(x-0.7)^2/sx^2-(z-0.8)^2/sz^2))
}

## create quadrature (integration) grid, in useful order
ig <- 5 ## integration grid within square
mx <- mz <- (1:ig-.5)/ig
ix <- rep(mx,ig);iz <- rep(mz,rep(ig,ig))

og <- 10 ## observarion grid
mx <- mz <- (1:og-1)/og
ox <- rep(mx,og);ox <- rep(ox,rep(ig^2,og^2))
oz <- rep(mz,rep(og,og));oz <- rep(oz,rep(ig^2,og^2))

x <- ox + ix/og;z <- oz + iz/og ## full grid, subsquare by subsquare

## create matrix covariates...
X <- matrix(x,og^2,ig^2,byrow=TRUE)
Z <- matrix(z,og^2,ig^2,byrow=TRUE)

## create simulated test data...
dA <- 1/(og*ig)^2 ## quadrature square area
F <- test1(X,Z) ## evaluate on grid

```

```

f <- rowSums(F)*dA ## integrate by midpoint quadrature
y <- f + rnorm(og^2)*5e-4 ## add noise
## ... so each y is a noisy observation of the integral of `test1'
## over a 0.1 by 0.1 sub-square from the unit square

## Now fit model to simulated data...

L <- X*0 + dA

## ... let F be the matrix of the smooth evaluated at the x,z values
## in matrices X and Z. rowSums(L*F) gives the model predicted
## integrals of `test1' corresponding to the observed `y'

L1 <- rowSums(L) ## smooths are centred --- need to add in L%*%1

## fit models to reconstruct `test1'....

b <- gam(y~s(X,Z,by=L)+L1-1) ## (L1 and const are confounded here)
b1 <- gam(y~te(X,Z,by=L)+L1-1) ## tensor product alternative

## plot results...

old.par<-par(mfrow=c(2,2))
x<-runif(n);z<-runif(n);
xs<-seq(0,1,length=30);zs<-seq(0,1,length=30)
pr<-data.frame(x=rep(xs,30),z=rep(zs,rep(30,30)))
truth<-matrix(test1(pr$x,pr$z),30,30)
contour(xs,zs,truth)
plot(b)
vis.gam(b,view=c("X","Z"),cond=list(L1=1,L=1),plot.type="contour")
vis.gam(b1,view=c("X","Z"),cond=list(L1=1,L=1),plot.type="contour")

#####
## A "signal" regression example...#
#####

rf <- function(x=seq(0,1,length=100)) {
## generates random functions...
  m <- ceiling(runif(1)*5) ## number of components
  f <- x*0;
  mu <- runif(m,min(x),max(x));sig <- (runif(m)+.5)*(max(x)-min(x))/10
  for (i in 1:m) f <- f+ dnorm(x,mu[i],sig[i])
  f
}

x <- seq(0,1,length=100) ## evaluation points

## example functional predictors...
par(mfrow=c(3,3));for (i in 1:9) plot(x,rf(x),type="l",xlab="x")

## simulate 200 functions and store in rows of L...
L <- matrix(NA,200,100)
for (i in 1:200) L[i,] <- rf() ## simulate the functional predictors

f2 <- function(x) { ## the coefficient function
  (0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10)/10
}

```

```

f <- f2(x) ## the true coefficient function

y <- L%*%f + rnorm(200)*20 ## simulated response data

## Now fit the model E(y) = L%*%f(x) where f is a smooth function.
## The summation convention is used to evaluate smooth at each value
## in matrix X to get matrix F, say. Then rowSum(L*F) gives E(y).

## create matrix of eval points for each function. Note that
## `smoothCon' is smart and will recognize the duplication...
X <- matrix(x,200,100,byrow=TRUE)

b <- gam(y~s(X,by=L,k=20))
par(mfrow=c(1,1))
plot(b,shade=TRUE);lines(x,f,col=2)

```

logLik.gam

gam[AIC](#)

```

## S3 method for class 'gam'
logLik(object,...)

```

```

object      gamgam()
...

```

logLik.glmgam
[AICgam](#)
[gaussiangelm](#)
mgcvMLREML[gambam](#)

logLik[logLik](#)

<simon.wood@r-project.org>logLik.glm

[AIC](#)

lp

\mathbf{x}

$$\min \mathbf{c}^T \mathbf{x} \mathbf{A} \mathbf{x} = \mathbf{b}, \mathbf{b} \geq \mathbf{0}$$

$$\min \mathbf{c}^T \mathbf{x} \mathbf{A} \mathbf{x} \geq \mathbf{b}, \mathbf{C} \mathbf{x} = \mathbf{d}$$

feasible

```
lp(c,A,b,C=NULL,d=NULL,Bi=NULL,maxit=max(1000, nrow(A) * 10), phase1 = FALSE)
feasible(A,b,C=NULL,d=NULL,maxit = max(1000, nrow(A) * 10))
```

\mathbf{c} $\mathbf{c}^T \mathbf{x}$

\mathbf{A}

\mathbf{b} \mathbf{A}

\mathbf{C} NULL

\mathbf{d} NULL

\mathbf{B}_i $\mathbf{x} \mathbf{A}[:, B_i] \mathbf{x}^* = \mathbf{b} \mathbf{x}^* \geq \mathbf{0} \text{NULL}$

maxit

phase1

$\mathbf{A}[:, B_i] \mathbf{B}_i \mathbf{A}[:, B_i]$

lpfeasiblemaxit

<simon.wood@r-project.org>

pcls

```

library(mgcv)
## very simple linear program...
c <- c(-4,-2,0,0)
A <- matrix(c(1,2,1,.5,1,0,0,1),2,4)
b <- c(5,8)
x <- lp(c,A,b,c(3,4));sum(c*x);x ## Bi given

x <- lp(c,A,b);sum(c*x);x ## Bi found automatically

## equivalent formulation Ax>b
A <- -matrix(c(1,2,1,.5),2,2)
b <- -c(5,8)
C <- matrix(0,0,2); d <- numeric(0)
c <- c(-4,-2)
x <- lp(c,A,b,C=C,d=d);sum(c*x);x

## equivalent formulation Ax>b, Cx=d
c <- c(-4,-2,0)
A <- matrix(c(0,-2,0,-.5,1,0),2,3)
C <- matrix(c(1,1,1),1,3); d <- 5
b <- c(0,-8)
x <- lp(c,A,b,C=C,d=d);sum(c*x);x

```

ls.size

```
ls.size(x)
```

```
x
```

```
xx
```

```
<simon.wood@r-project.org>
```

```

library(mgcv)
b <- list(M=matrix(runif(100),10,10),quote=
"The world is ruled by idiots because only an idiot would want to rule the world.",
fam=binomial())
ls.size(b)

```

magic

```
magic(y,X,sp,S,off,L=NULL,lsp0=NULL,rank=NULL,H=NULL,C=NULL,
      w=NULL,gamma=1,scale=1,gcv=TRUE,ridge.parameter=NULL,
      control=list(tol=1e-6,step.half=25,rank.tol=
        .Machine$double.eps^0.5),extra.rss=0,n.score=length(y),nthreads=1)
```

y

X

sp L%%log(sp) + lsp0SNULLsp

S S[[i]]S[[i]]off[i]off[i]S[[i]]list()

off S[[i]]

L log(sp)SNULLsp

lsp0 LNULLlog(sp)S[[i]]L%%log(sp) + lsp0NULL

rank

H

C $\mathbf{bCb} = \mathbf{0}$

w $y\mathbf{V}_y^{-1} = \mathbf{w}'\mathbf{w}y$

gamma

scale

gcv TRUEFALSE

ridge.parameter

control

step.half

rank_tol

extra.rss n.score

n.score

nthreads magic

$$minimise \| \mathbf{W}(\mathbf{X}\mathbf{b} - \mathbf{y}) \|^2 + \mathbf{b}'\mathbf{H}\mathbf{b} + \sum_{i=1}^m \theta_i \mathbf{b}'\mathbf{S}_i \mathbf{b}$$

$$\mathbf{C}\mathbf{b} = \mathbf{0}\mathbf{X}\mathbf{b}\mathbf{y}\mathbf{W}\mathbf{S}_i\theta_i\mathbf{H}\mathbf{C}\mathbf{X}$$

$$\theta_i$$

$$V_g = \frac{n\|\mathbf{W}(\mathbf{y} - \mathbf{A}\mathbf{y})\|^2}{[tr(\mathbf{I} - \gamma\mathbf{A})]^2}$$

$$V_u = \|\mathbf{W}(\mathbf{y} - \mathbf{A}\mathbf{y})\|^2/n - 2\phi tr(\mathbf{I} - \gamma\mathbf{A})/n + \phi$$

$$\gamma\mathrm{gamma}\phi\mathrm{scale}\mathbf{A}\mathbf{A}$$

$$\theta_i\theta_i\log(\theta_i)\mathbf{b}$$

$$\mathbf{C}$$

$$\mathbf{b}$$

$$\mathrm{scale}$$

$$\mathrm{score}$$

$$\mathrm{sp}$$

$$\mathrm{sp.full}\qquad\mathrm{SspLNULLexp(L\%\%log(sp))}$$

$$\mathrm{rV}\qquad\mathrm{brV\%\%t(rV)*scale}$$

$$\mathrm{gcv.info}$$

$$\mathrm{TRUEFALSE}$$

$$\mathrm{TRUE}$$

$$\mathbf{X}$$

$$\mathrm{magic.post.proc}$$

$$<\mathrm{simon.wood@r-project.org}>$$

$$\mathrm{magic.post.progam}$$

```

## Use 'magic' for a standard additive model fit ...
library(mgcv)
set.seed(1); n <- 200; sig <- 1
dat <- gamSim(1, n=n, scale=sig)
k <- 30
## set up additive model
G <- gam(y~s(x0,k=k)+s(x1,k=k)+s(x2,k=k)+s(x3,k=k), fit=FALSE, data=dat)
## fit using magic (and gam default tolerance)
mgfit <- magic(G$y, G$X, G$sp, G$S, G$off, rank=G$rank,
               control=list(tol=1e-7, step.half=15))
## and fit using gam as consistency check
b <- gam(G=G)
mgfit$sp; b$sp # compare smoothing parameter estimates
edf <- magic.post.proc(G$X, mgfit, G$w)$edf # get e.d.f. per param
range(edf-b$edf) # compare

## p>n example... fit model to first 100 data only, so more
## params than data...

mgfit <- magic(G$y[1:100], G$X[1:100,], G$sp, G$S, G$off, rank=G$rank)
edf <- magic.post.proc(G$X[1:100,], mgfit, G$w[1:100])$edf

## constrain first two smooths to have identical smoothing parameters
L <- diag(3); L <- rbind(L[1,], L)
mgfit <- magic(G$y, G$X, rep(-1, 3), G$S, G$off, L=L, rank=G$rank, C=G$C)

## Now a correlated data example ...
library(nlme)
## simulate truth
set.seed(1); n<-400; sig<-2
x <- 0:(n-1)/(n-1)
f <- 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
## produce scaled covariance matrix for AR1 errors...
V <- corMatrix(Initialize(corAR1(.6), data.frame(x=x)))
Cv <- chol(V) # t(Cv)%*%Cv=V
## Simulate AR1 errors ...
e <- t(Cv)%*%rnorm(n, 0, sig) # so cov(e) = V * sig^2
## Observe truth + AR1 errors
y <- f + e
## GAM ignoring correlation
par(mfrow=c(1,2))
b <- gam(y~s(x, k=20))
plot(b); lines(x, f-mean(f), col=2); title("Ignoring correlation")
## Fit smooth, taking account of *known* correlation...
w <- solve(t(Cv)) # V^{-1} = w'w
## Use 'gam' to set up model for fitting...
G <- gam(y~s(x, k=20), fit=FALSE)
## fit using magic, with weight *matrix*
mgfit <- magic(G$y, G$X, G$sp, G$S, G$off, rank=G$rank, C=G$C, w=w)
## Modify previous gam object using new fit, for plotting...
mg.stuff <- magic.post.proc(G$X, mgfit, w)
b$edf <- mg.stuff$edf; b$Vp <- mg.stuff$Vb
b$coefficients <- mgfit$b
plot(b); lines(x, f-mean(f), col=2); title("Known correlation")

```

magic.post.proc

magic

magic.post.proc(X,object,w=NULL)

X
object magicX
w magicwt(w)%*%wmagic

$\text{objectrVscale}\phi\mathbf{V}\mathbf{V}'\phi\mathbf{V}\mathbf{V}'\mathbf{X}'\mathbf{W}'\mathbf{W}\mathbf{X}\mathbf{W}\text{diag}(w)\mathbf{W}\mathbf{X}\mathbf{V}\mathbf{V}'\mathbf{X}'\mathbf{W}'$
 $\mathbf{V}\mathbf{V}'\mathbf{X}'\mathbf{W}'\mathbf{W}\mathbf{X}\mathbf{V}\mathbf{V}'\phi$

Vb
Ve
hat
edf

<simon.wood@r-project.org>

magic

mchol	chol
-------	------

CholeskyMatrixMatrixchol

mchol(A)

A CholeskyMatrix

Matrixcholexpand1chol(A,pivot=TRUE)Amatrix

A"pivot""rank"A-1"rank"-1

<simon.wood@r-project.org>

```
library(mgcv)
## A sparse +ve def matrix
u <- sample(1:100,10)*.001
x <- i <- j <- 1:10
ii <- sample(1:10,10,replace=TRUE);
jj <- sample(1:10,10,replace=TRUE)
x <- c(x,u,u);
i <- c(i,ii,jj)
j <- c(j,jj,ii)
A <- Matrix::sparseMatrix(i=i,j=j,x=x)
R <- mchol(A)
piv <- attr(R,"pivot")
range(crossprod(R)-A[piv,piv])

i <- sample(1:5,10,replace=TRUE)
j <- sample(1:10,10,replace=TRUE)
u <- sample(1:100,10)*.001
A <- crossprod(Matrix::sparseMatrix(i=i,j=j,x=u))
mchol(A)
```

mgcv.FAQ

[summary.gamgam4](#)

[predict.gam](#)

[tprs](#)

[multinommultinom](#)

[gam.methodmethodgamo](#)[optimizer](#)[gam](#)

[gamgam4](#)[gamlmer](#)

"tp"

<simon.wood@r-project.org>

mgcv.package

mgcv**gam**gam**random**.effectsscasmfamily.mgcvsmooth.terms
jagamginla

mgcv**gam**predict.gamplot.gamsummary.gamanova.gamgamObject
gamglm**gam**stetit2smooth.termsrandom.effectsmrfNCVgamgam.models
linear.functional.term**gam**.selectiongam.convergencegammethodoptimizer
gam.controlgam.checkchoose.kvis.gamplot.gamsmooth.construct
V**gam**Objectpredict.gamanova.gamsummary.gam
bamgambam(...,discrete=TRUE)
gammlmenlme4**gam**4**gam**random.effects**gam**bamscasm
magic
magic**gam**pcls
library(help=mgcv)mgcv.FAQ

<https://webhomes.maths.ed.ac.uk/~swood34/>

see examples for gam, bam and gamm

mgcv.parallel

```
mgcv
bamdiscrete=TRUEbamnthreads
bam
gamopenMPnthreadscontrolgam $O(np^2)O(p^3)$ magic
NCVgamncv.threadsgam.control
control$nthreads
```

mgcv

<https://hpc-tutorials.llnl.gov/openmp/>

```
## illustration of multi-threading with gam...

require(mgcv);set.seed(9)
dat <- gamSim(1,n=2000,dist="poisson",scale=.1)
k <- 12;bs <- "cr";ctrl <- list(nthreads=2)

system.time(b1<-gam(y~s(x0,bs=bs)+s(x1,bs=bs)+s(x2,bs=bs,k=k),
  ,family=poisson,data=dat,method="REML"))[3]

system.time(b2<-gam(y~s(x0,bs=bs)+s(x1,bs=bs)+s(x2,bs=bs,k=k),
  family=poisson,data=dat,method="REML",control=ctrl))[3]

## Poisson example on a cluster with 'bam'.
## Note that there is some overhead in initializing the
## computation on the cluster, associated with loading
## the Matrix package on each node. Sample sizes are low
## here to keep example quick -- for such a small model
## little or no advantage is likely to be seen.
k <- 13;set.seed(9)
dat <- gamSim(1,n=6000,dist="poisson",scale=.1)

require(parallel)
nc <- 2 ## cluster size, set for example portability
if (detectCores()>1) { ## no point otherwise
  cl <- makeCluster(nc)
  ## could also use makeForkCluster, but read warnings first!
} else cl <- NULL
```

```

system.time(b3 <- bam(y ~ s(x0,bs=bs,k=7)+s(x1,bs=bs,k=7)+s(x2,bs=bs,k=k)
, data=dat, family=poisson(), chunk.size=5000, cluster=c1))

fv <- predict(b3, cluster=c1) ## parallel prediction

if (!is.null(c1)) stopCluster(c1)
b3

## Alternative, better scaling example, using the discrete option with bam...

system.time(b4 <- bam(y ~ s(x0,bs=bs,k=7)+s(x1,bs=bs,k=7)+s(x2,bs=bs,k=k)
, data=dat, family=poisson(), discrete=TRUE, nthreads=2))

```

mini.roots

```

B[[i]]S[[i]]

mini.roots(S, off, np, rank = NULL)

S
off          1:off[i]B[[i]]
np
rank          rank[i]S[[i]]rank[i] < 1rank=NULL

S[[i]]=B[[i]]%*%t(B[[i]])

```

missing.data

```

bys(x)xgam.modelsby

```

```

gam.vcompgam.modelssmooth.construct.re.smooth.specgam

```

```

## The example takes a couple of minutes to run...

require(mgcv)
par(mfrow=c(4,4),mar=c(4,4,1,1))
for (sim in c(1,7)) { ## cycle over uncorrelated and correlated covariates
  n <- 350;set.seed(2)
  ## simulate data but randomly drop 300 covariate measurements
  ## leaving only 50 complete cases...
  dat <- gamSim(sim,n=n,scale=3) ## 1 or 7
  drop <- sample(1:n,300) ## to
  for (i in 2:5) dat[drop[1:75+(i-2)*75],i] <- NA

  ## process data.frame producing binary indicators of missingness,
  ## mx0, mx1 etc. For each missing value create a level of a factor
  ## idx0, idx1, etc. So idx0 has as many levels as x0 has missing
  ## values. Replace the NA's in each variable by the mean of the
  ## non missing for that variable...

  dname <- names(dat)[2:5]
  dat1 <- dat
  for (i in 1:4) {
    by.name <- paste("m",dname[i],sep="")
    dat1[[by.name]] <- is.na(dat1[[dname[i]]])
    dat1[[dname[i]]][dat1[[by.name]]] <- mean(dat1[[dname[i]]],na.rm=TRUE)
    lev <- rep(1,n);lev[dat1[[by.name]]] <- 1:sum(dat1[[by.name]])
    id.name <- paste("id",dname[i],sep="")
    dat1[[id.name]] <- factor(lev)
    dat1[[by.name]] <- as.numeric(dat1[[by.name]])
  }

  ## Fit a gam, in which any missing value contributes zero
  ## to the linear predictor from its smooth, but each
  ## missing has its own random effect, with the random effect
  ## variances being specific to the variable. e.g.
  ## for s(x0,by=ordered(!mx0)), declaring the `by' as an ordered
  ## factor ensures that the smooth is centred, but multiplied
  ## by zero when mx0 is one (indicating a missing x0). This means
  ## that any value (within range) can be put in place of the
  ## NA for x0. s(idx0,bs="re",by=mx0) produces a separate Gaussian
  ## random effect for each missing value of x0 (in place of s(x0),
  ## effectively). The `by' variable simply sets the random effect to
  ## zero when x0 is non-missing, so that we can set idx0 to any
  ## existing level for these cases.

  b <- bam(y~s(x0,by=ordered(!mx0))+s(x1,by=ordered(!mx1))+
    s(x2,by=ordered(!mx2))+s(x3,by=ordered(!mx3))+
    s(idx0,bs="re",by=mx0)+s(idx1,bs="re",by=mx1)+
    s(idx2,bs="re",by=mx2)+s(idx3,bs="re",by=mx3)
    ,data=dat1,discrete=TRUE)

  for (i in 1:4) plot(b,select=i) ## plot the smooth effects from b

  ## fit the model to the `complete case' data...
  b2 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="REML")
  plot(b2) ## plot the complete case results
}

```

```
model.matrix.gam
```

```
gam
```

```
## S3 method for class 'gam'  
model.matrix(object, ...)
```

```
object      gamgam()  
...         predict.gam
```

```
predict.gamnewdatatype="lpmatrix"object
```

```
<simon.wood@r-project.org>
```

```
gam
```

```
require(mgcv)  
n <- 15  
x <- runif(n)  
y <- sin(x*2*pi) + rnorm(n)*.2  
mod <- gam(y~s(x,bs="cc",k=6),knots=list(x=seq(0,1,length=6)))  
model.matrix(mod)
```

mono.con

gam"cr"

mono.con(x,up=TRUE,lower=NA,upper=NA)

x
up TRUEFALSE
lower NA
upper NA

$\{x_i, p_i : i = 1 \dots n\}$ **pAp ≥ b**

Ab

<simon.wood@r-project.org>

magicpcls

see ?pcls

mroot

mroot(A,rank=NULL,method="chol")

A
rank ANULL
method "chol""svd"

A

$$\mathbf{BAA} = \mathbf{BB}'$$

<simon.wood@r-project.org>

```
require(mgcv)
set.seed(0)
a <- matrix(runif(24),6,4)
A <- a%*%t(a) ## A is +ve semi-definite, rank 4
B <- mroot(A) ## default pivoted choleski method
tol <- 100*.Machine$double.eps
chol.err <- max(abs(A-B%*%t(B)));chol.err
if (chol.err>tol) warning("mroot (chol) suspect")
B <- mroot(A,method="svd") ## svd method
svd.err <- max(abs(A-B%*%t(B)));svd.err
if (svd.err>tol) warning("mroot (svd) suspect")
```

multinom

[gamgamformula.gam](#)

multinom(K=1)

K

$$\eta_j \exp(\eta_j) / \{1 + \sum_j \exp(\eta_j)\} 1 / \{1 + \sum_j \exp(\eta_j)\}$$

predicttype="response"

general.family

<simon.wood@r-project.org>

ocat

```
library(mgcv)
set.seed(6)
## simulate some data from a three class model
n <- 1000
f1 <- function(x) sin(3*pi*x)*exp(-x)
f2 <- function(x) x^3
f3 <- function(x) .5*exp(-x^2)-.2
f4 <- function(x) 1
x1 <- runif(n);x2 <- runif(n)
eta1 <- 2*(f1(x1) + f2(x2))-.5
eta2 <- 2*(f3(x1) + f4(x2))-1
p <- exp(cbind(0,eta1,eta2))
p <- p/rowSums(p) ## prob. of each category
cp <- t(apply(p,1,cumsum)) ## cumulative prob.
## simulate multinomial response with these probabilities
## see also ?rmultinom
y <- apply(cp,1,function(x) min(which(x>runif(1))))-1
## plot simulated data...
plot(x1,x2,col=y+3)

## now fit the model...
b <- gam(list(y~s(x1)+s(x2),~s(x1)+s(x2)),family=multinom(K=2))
plot(b,pages=1)
gam.check(b)

## now a simple classification plot...
expand.grid(x1=seq(0,1,length=40),x2=seq(0,1,length=40)) -> gr
pp <- predict(b,newdata=gr,type="response")
pc <- apply(pp,1,function(x) which(max(x)==x)[1])-1
plot(gr,col=pc+3,pch=19)

## example sharing a smoother between linear predictors
## ?formula.gam gives more details.
b <- gam(list(y~s(x1),~s(x1),1+2~s(x2)-1),family=multinom(K=2))
plot(b,pages=1)
```

mvn

gamformula.gam

mvn(d=2)

d

d
"response""deviance"

general.family

<simon.wood@r-project.org>

gaussian

```
library(mgcv)
## simulate some data...
V <- matrix(c(2,1,1,2),2,2)
f0 <- function(x) 2 * sin(pi * x)
f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
      (10 * x)^3 * (1 - x)^10
n <- 300
x0 <- runif(n); x1 <- runif(n);
x2 <- runif(n); x3 <- runif(n)
y <- matrix(0,n,2)
for (i in 1:n) {
  mu <- c(f0(x0[i])+f1(x1[i]),f2(x2[i]))
  y[i,] <- rmvn(1,mu,V)
}
dat <- data.frame(y0=y[,1],y1=y[,2],x0=x0,x1=x1,x2=x2,x3=x3)

## fit model...

b <- gam(list(y0~s(x0)+s(x1),y1~s(x2)+s(x3)),family=mvn(d=2),data=dat)
b
summary(b)
plot(b,pages=1)
solve(crossprod(b$family$data$R)) ## estimated cov matrix
```


$$\sum_i D(y_i,\theta_i) + \sum_j \lambda_j \beta^\mathsf{T} S_j \beta$$

$$Dy_i\theta_i\beta S_j\lambda_j\beta$$

$$k=1,\ldots,m\alpha(k)\subset\{1,\ldots,n\}\delta(k)\subset\{1,\ldots,n\}\delta(k)\alpha(k)\theta_i^{\alpha(k)}\theta_i\alpha(k)$$

$$V=\sum_{k=1}^m\sum_{i\in\delta(k)}D(y_i,\theta_i^{\alpha(k)})$$

$$\lambda_jm=n\alpha(k)=\delta(k)=k\\ VO(n^{-2})\beta\alpha(k)\mathrm{mgcv}O(p^2)p\beta O(np^2)$$

$$\mathrm{bam}(\ldots,\mathrm{discrete}=\mathrm{TRUE})\mathrm{samplenei}$$

$$m=n\delta(k)=k\alpha(k)=\mathrm{nei}(k)\mathrm{nei}(\mathbf{k})k$$

$$\alpha(k)\delta(k)\textcolor{blue}{gam}\mathrm{nei}$$

$$\begin{array}{l} \mathbf{a} \\ \mathrm{manei}\$ \mathbf{a}[(\mathrm{nei}\$ \mathbf{ma}[\mathbf{j}-1]+1):\mathrm{nei}\$ \mathbf{ma}[\mathbf{j}]] \mathbf{j} \alpha(j) \\ \mathbf{d} \\ \mathrm{mdmdnei}\$ \mathbf{d}[(\mathrm{nei}\$ \mathbf{md}[\mathbf{j}-1]+1):\mathrm{nei}\$ \mathbf{md}[\mathbf{j}]] \delta(j) \\ \mathrm{sample}\textcolor{blue}{bam} \\ \mathrm{jackknife}\textcolor{blue}{gam}\mathrm{TRUE}\mathrm{FALSE}\mathrm{jackknife} \\ \mathrm{nei}==\mathrm{NULL}\mathrm{amanei} \end{array}$$

$$\textcolor{blue}{\beta}\textcolor{blue}{gevlss}\textcolor{blue}{bam} \\ \mathrm{ncv}.\mathrm{threads}\textcolor{blue}{gam}.\textcolor{blue}{control}$$

$$<\mathrm{simon.wood}@r\text{-}project.org>$$

$$\textcolor{red}{\mathrm{https://arxiv.org/abs/2404.16490}}$$

```

require(mgcv)
nei.cor <- function(h,n) { ## construct nei structure
  nei <- list(md=1:n,d=1:n)
  nei$ma <- cumsum(c((h+1):(2*h+1),rep(2*h+1,n-2*h-2),(2*h+1):(h+1)))
  a0 <- rep(0,0); if (h>0) for (i in 1:h) a0 <- c(a0,1:(h+i))
  a1 <- n-a0[length(a0):1]+1
  nei$a <- c(a0,1:(2*h+1)+rep(0:(n-2*h-1),each=2*h+1),a1)
  nei
}
set.seed(1)
n <- 500;sig <- .6
x <- 0:(n-1)/(n-1)
f <- sin(4*pi*x)*exp(-x*2)*5/2
e <- rnorm(n,0,sig)
for (i in 2:n) e[i] <- 0.6*e[i-1] + e[i]
y <- f + e ## autocorrelated data
nei <- nei.cor(4,n) ## construct neighbourhoods to mitigate
b0 <- gam(y~s(x,k=40)) ## GCV based fit
gc <- gam.control(ncv.threads=2)
b1 <- gam(y~s(x,k=40),method="NCV",nei=nei,control=gc)
## use "QNCV", which is identical here...
b2 <- gam(y~s(x,k=40),method="QNCV",nei=nei,control=gc)
## plot GCV and NCV based fits...
f <- f - mean(f)
par(mfrow=c(1,2))
plot(b0,rug=FALSE,scheme=1);lines(x,f,col=2)
plot(b1,rug=FALSE,scheme=1);lines(x,f,col=2)

```

negbin

gam**negbin**negative.binomial**nb**theta θ $var(y) = \mu + \mu^2/\theta$ $\mu = E(y)$

theta**gam**

negbin**gam**theta

nbtheta

optimizer**gam**"perf"

negbin(theta = stop("'theta' must be specified"), link = "log")

nb(theta = NULL, link = "log")

theta negbinnbtheta**theta**

link "log""identity""sqrt"

nbtheta**gam****gam**

negbintheta**gam****gam**theta[2]>theta[1]**gam**optimizer $\hat{\theta}\hat{\theta}\hat{\theta}\hat{\theta}$

```
negbinfamily
dvar          mu
d2var         mu
getTheta      theta
nbextended.family
```

```
gammtheta
```

```
<simon.wood@r-project.org>negative.binomial
```

```
library(mgcv)
set.seed(3)
n<-400
dat <- gamSim(1,n=n)
g <- exp(dat$f/5)

## negative binomial data...
dat$y <- rnbinom(g,size=3,mu=g)
## known theta fit ...
b0 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=negbin(3),data=dat)
plot(b0,pages=1)
print(b0)

## same with theta estimation...
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=nb(),data=dat)
plot(b,pages=1)
print(b)
b$family$getTheta(TRUE) ## extract final theta estimate

## another example...
set.seed(1)
f <- dat$f
f <- f - min(f)+5;g <- f^2/10
dat$y <- rnbinom(g,size=3,mu=g)
b2 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=nb(link="sqrt"),
          data=dat,method="REML")
plot(b2,pages=1)
print(b2)
rm(dat)
```

new.name

[gammlme](#)

new.name(proposed,old.names)

proposed
old.names

old.names

<simon.wood@r-project.org>

[gamm](#)

require(mgcv)
old <- c("a","tuba","is","tubby")
new.name("tubby",old)

notExp

aa=exp(b)bbba
notExpexp
notLognotExp
pdMatlme[gamm](#)[notExp2](#)[notLog2](#)

notExp(x)
notLog(x)

x notExpnotLog

<simon.wood@r-project.org>

[pdTenspdIdnotgamm](#)

```
## Illustrate the notExp function:
## less steep than exp, but still monotonic.
require(mgcv)
x <- -100:100/10
op <- par(mfrow=c(2,2))
plot(x,notExp(x),type="l")
lines(x,exp(x),col=2)
plot(x,log(notExp(x)),type="l")
lines(x,log(exp(x)),col=2) # redundancy intended
x <- x/4
plot(x,notExp(x),type="l")
lines(x,exp(x),col=2)
plot(x,log(notExp(x)),type="l")
lines(x,log(exp(x)),col=2) # redundancy intended
par(op)
range(notLog(notExp(x))-x) # show that inverse works!
```

notExp2

notLog2notExp2logexp[notLognotExp](#)[pdTenspdIdnotgamm](#)

notExp2notLog2notExp2

notExp2

lmegamm

lmenotExp

notExp2(x,d=.Options\$mgcv.vc.logrange,b=1/d)

notLog2(x,d=.Options\$mgcv.vc.logrange,b=1/d)

x notExpnotLog

d notExp2exp(-d)exp(d)gammmgcv.vc.logrange[options](#)

b notExp2

<simon.wood@r-project.org>

pdTenspdIdnotgamm

```
## Illustrate the notExp2 function:
require(mgcv)
x <- seq(-50,50,length=1000)
op <- par(mfrow=c(2,2))
plot(x,notExp2(x),type="l")
lines(x,exp(x),col=2)
plot(x,log(notExp2(x)),type="l")
lines(x,log(exp(x)),col=2) # redundancy intended
x <- x/4
plot(x,notExp2(x),type="l")
lines(x,exp(x),col=2)
plot(x,log(notExp2(x)),type="l")
lines(x,log(exp(x)),col=2) # redundancy intended
par(op)
```

null.space.dimension

null.space.dimension $M d m m 2 m > d d$

null.space.dimension(d,m)

d
m

$m 2 m > d + 1 2 m < d + 1 m d m$
 $M = (m + d - 1)! / (d! (m - 1)!)$

M

<simon.wood@r-project.org>

tprs

```
require(mgcv)
null.space.dimension(2,0)
```

ocat

[gambam](#)

```
ocat(theta=NULL,link="identity",R=NULL)
```

```
theta      R=2
link       "identity"
R
```

```
ocatpredict.gamtype="response"
```

```
extended.family
```

```
<simon.wood@r-project.org>
```

```
library(mgcv)
## Simulate some ordered categorical data...
set.seed(3);n<-400
dat <- gamSim(1,n=n)
dat$f <- dat$f - mean(dat$f)

alpha <- c(-Inf,-1,0,5,Inf)
R <- length(alpha)-1
y <- dat$f
u <- runif(n)
u <- dat$f + log(u/(1-u))
for (i in 1:R) {
  y[u > alpha[i]&u <= alpha[i+1]] <- i
}
dat$y <- y

## plot the data...
par(mfrow=c(2,2))
with(dat,plot(x0,y));with(dat,plot(x1,y))
with(dat,plot(x2,y));with(dat,plot(x3,y))

## fit ocat model to data...
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=ocat(R=R),data=dat)
b
```

```

plot(b,pages=1)
gam.check(b)
summary(b)
b$family$getTheta(TRUE) ## the estimated cut points

## predict probabilities of being in each category
predict(b,dat[1:2,],type="response",se=TRUE)

```

one.se.rule

$$\rho \rho^T N(\rho, V) V \text{sp.vcov} \rho V V d \alpha d \alpha \alpha d^T V^{-1} d = \sqrt{2p}$$

<simon.wood@r-project.org>

gam

```

require(mgcv)
set.seed(2) ## simulate some data...
dat <- gamSim(1,n=400,dist="normal",scale=2)
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="REML")
b
## only the first 3 smoothing parameters are candidates for
## increasing here...
V <- sp.vcov(b)[1:3,1:3] ## the approx cov matrix of sps
d <- diag(V)^.5          ## sp se.
## compute the log smoothing parameter step...
d <- sqrt(2*length(d))/d
sp <- b$sp ## extract original sp estimates
sp[1:3] <- sp[1:3]*exp(d) ## apply the step
## refit with the increased smoothing parameters...
b1 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="REML",sp=sp)
b;b1 ## compare fits

```

pcls

pcls(M)

M pcls

ncol(M\$X)nrow(M\$X)
CCp = ccp
S[[i]]off[i]+1
M\$S

active
A_{in}p > b_{in}

pclsp

$$minimise \|W^{1/2}(\mathbf{X}\mathbf{p} - \mathbf{y})\|^2 + \sum_{i=1}^m \lambda_i \mathbf{p}' \mathbf{S}_i \mathbf{p}$$

Cp = cA_{in}p > b_{in}pλ_iXpyWS_iCλ_iXA_{in}b_{in}
X'X

active

<simon.wood@r-project.org>

magicmono.com

```

require(mgcv)
# first an un-penalized example - fit  $E(y)=a+bx$  subject to  $a>0$ 
set.seed(0)
n <- 100
x <- runif(n); y <- x - 0.2 + rnorm(n)*0.1
M <- list(X=matrix(0,n,2),p=c(0.1,0.5),off=array(0,0),S=list(),
Ain=matrix(0,1,2),bin=0,C=matrix(0,0,0),sp=array(0,0),y=y,w=y*0+1)
M$X[,1] <- 1; M$X[,2] <- x; M$Ain[1,] <- c(1,0)
pcls(M) -> M$p
plot(x,y); abline(M$p,col=2); abline(coef(lm(y~x)),col=3)

# Penalized example: monotonic penalized regression spline ....

# Generate data from a monotonic truth.
x <- runif(100)*4-1;x <- sort(x);
f <- exp(4*x)/(1+exp(4*x)); y <- f+rnorm(100)*0.1; plot(x,y)
dat <- data.frame(x=x,y=y)
# Show regular spline fit (and save fitted object)
f.ug <- gam(y~s(x,k=10,bs="cr")); lines(x,fitted(f.ug))
# Create Design matrix, constraints etc. for monotonic spline....
sm <- smoothCon(s(x,k=10,bs="cr"),dat,knots=NULL)[[1]]
F <- mono.con(sm$xp); # get constraints
G <- list(X=sm$X,C=matrix(0,0,0),sp=f.ug$sp,p=sm$xp,y=y,w=y*0+1)
G$Ain <- F$A;G$bin <- F$b;G$S <- sm$S;G$off <- 0

p <- pcls(G); # fit spline (using s.p. from unconstrained fit)

fv<-Predict.matrix(sm,data.frame(x=x))%*%p
lines(x,fv,col=2)

# now a tprs example of the same thing...

f.ug <- gam(y~s(x,k=10)); lines(x,fitted(f.ug))
# Create Design matrix, constraints etc. for monotonic spline....
sm <- smoothCon(s(x,k=10,bs="tp"),dat,knots=NULL)[[1]]
xc <- 0:39/39 # points on [0,1]
nc <- length(xc) # number of constraints
xc <- xc*4-1 # points at which to impose constraints
A0 <- Predict.matrix(sm,data.frame(x=xc))
# ... A0%*%p evaluates spline at xc points
A1 <- Predict.matrix(sm,data.frame(x=xc+1e-6))
A <- (A1-A0)/1e-6
## ... approx. constraint matrix (A%*%p is -ve
## spline gradient at points xc)
G <- list(X=sm$X,C=matrix(0,0,0),sp=f.ug$sp,y=y,w=y*0+1,S=sm$S,off=0)
G$Ain <- A; # constraint matrix
G$bin <- rep(0,nc); # constraint vector
G$p <- rep(0,10); G$p[10] <- 0.1
# ... monotonic start params, got by setting coefs of polynomial part
p <- pcls(G); # fit spline (using s.p. from unconstrained fit)

fv2 <- Predict.matrix(sm,data.frame(x=x))%*%p
lines(x,fv2,col=3)

#####
## monotonic additive model example...

```

```
#####

## First simulate data...

set.seed(10)
f1 <- function(x) 5*exp(4*x)/(1+exp(4*x));
f2 <- function(x) {
  ind <- x > .5
  f <- x*0
  f[ind] <- (x[ind] - .5)^2*10
  f
}
f3 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 +
  10 * (10 * x)^3 * (1 - x)^10
n <- 200
x <- runif(n); z <- runif(n); v <- runif(n)
mu <- f1(x) + f2(z) + f3(v)
y <- mu + rnorm(n)

## Preliminary unconstrained gam fit...
G <- gam(y~s(x)+s(z)+s(v,k=20),fit=FALSE)
b <- gam(G=G)

## generate constraints, by finite differencing
## using predict.gam ....
eps <- 1e-7
pd0 <- data.frame(x=seq(0,1,length=100),z=rep(.5,100),
  v=rep(.5,100))
pd1 <- data.frame(x=seq(0,1,length=100)+eps,z=rep(.5,100),
  v=rep(.5,100))
X0 <- predict(b,newdata=pd0,type="lpmatrix")
X1 <- predict(b,newdata=pd1,type="lpmatrix")
Xx <- (X1 - X0)/eps ## Xx %*% coef(b) must be positive
pd0 <- data.frame(z=seq(0,1,length=100),x=rep(.5,100),
  v=rep(.5,100))
pd1 <- data.frame(z=seq(0,1,length=100)+eps,x=rep(.5,100),
  v=rep(.5,100))
X0 <- predict(b,newdata=pd0,type="lpmatrix")
X1 <- predict(b,newdata=pd1,type="lpmatrix")
Xz <- (X1-X0)/eps
G$Ain <- rbind(Xx,Xz) ## inequality constraint matrix
G$bin <- rep(0,nrow(G$Ain))
G$C = matrix(0,0,ncol(G$X))
G$sp <- b$sp
G$p <- coef(b)
G$off <- G$off-1 ## to match what pcls is expecting
## force initial parameters to meet constraint
G$p[11:18] <- G$p[2:9]<- 0
p <- pcls(G) ## constrained fit
par(mfrow=c(2,3))
plot(b) ## original fit
b$coefficients <- p
plot(b) ## constrained fit
## note that standard errors in preceding plot are obtained from
## unconstrained fit
```

pdIdnot

pdMatpdIdentnlmepdMat[notLog2pdTens](#)

[pdTens](#)

```
pdIdnot(value = numeric(0), form = NULL,  
        nam = NULL, data = sys.frame(sys.parent()))
```

value

form

nam

data

Dim.pdIndotcoef.pdIdnotcorMatrix.pdIdnotlogDet.pdIdnotpdConstruct.pdIdnot
pdFactor.pdIdnotpdMatrix.pdIdnotsolve.pdIdnotsummary.pdIdnotmgcv:::coef.pdIdnot
pdFactorpdMatrixpdConstruct

pdIdnotnlme

<simon.wood@r-project.org>

nlme

[tepdTensnotLog2gamm](#)

see gamm

pdTens

nlmepdMatlmegammpdMatpdTens[notLog2](#)

pdTenspdConstruct.pdTenspdFactor.pdTenspdMatrix.pdTenscoef.pdTenssummary.pdTens

```
pdTens(value = numeric(0), form = NULL,  
       nam = NULL, data = sys.frame(sys.parent()))
```

value

form S

nam

data

Slmes[smooth.construct.tensor.smooth.spec](#)

pdMat

pdFactorpdMatrixpdConstruct

pdTenspdFactorpdMatrix

<simon.wood@r-project.org>

nlme

[tegamm](#)

see gamm

pen.edf

t2te

pen.edf(x)

x gam

t2

<simon.wood@r-project.org>

t2

```
require(mgcv)
set.seed(20)
dat <- gamSim(1,n=400,scale=2) ## simulate data
## following `t2' smooth basically separates smooth
## of x0,x1 into main effects + interaction....

b <- gam(y~t2(x0,x1,bs="tp",m=1,k=7)+s(x2)+s(x3),
         data=dat,method="ML")
pen.edf(b)

## label "rr" indicates interaction edf (range space times range space)
## label "nr" (null space for x0 times range space for x1) is main
##      effect for x1.
## label "rn" is main effect for x0
## clearly interaction is negligible

## second example with higher order marginals.

b <- gam(y~t2(x0,x1,bs="tp",m=2,k=7,full=TRUE)
         +s(x2)+s(x3),data=dat,method="ML")
pen.edf(b)

## In this case the EDF is negligible for all terms in the t2 smooth
## apart from the `main effects' (r2 and 2r). To understand the labels
## consider the following 2 examples....
## "r1" relates to the interaction of the range space of the first
##      marginal smooth and the first basis function of the null
##      space of the second marginal smooth
## "2r" relates to the interaction of the second basis function of
##      the null space of the first marginal smooth with the range
##      space of the second marginal smooth.
```

place.knots

```
place.knots(x,nk)
```

x

nk

<simon.wood@r-project.org>

[smooth.construct.cc.smooth.spec](#)

```
require(mgcv)
x<-runif(30)
place.knots(x,7)
rm(x)
```

plot.gam

```
gamgam()
```

```
## S3 method for class 'gam'
plot(x,residuals=FALSE,rug=NULL,se=TRUE,pages=0,select=NULL,scale=-1,
      n=100,n2=40,n3=3,theta=30,phi=30,jit=FALSE,xlab=NULL,
      ylab=NULL,main=NULL,ylim=NULL,xlim=NULL,too.far=0.1,
      all.terms=FALSE,shade.col="gray80",shift=0,
      trans=I,seWithMean=FALSE,unconditional=FALSE,by.resids=FALSE,
      scheme=0,deriv=FALSE,...)
```

```

x          gamgam()
residuals  TRUEFALSETRUE
rug        TRUENULLrugTRUEFALSE
se         TRUEFALSE
pages      pages=1
select     select=2
scale      ylim
n
n2
n3
theta
phi
jit
xlab
ylab
main
ylim
xlim
too.far    too.far
all.terms  TRUEtermplot
shade.col  rgb
shift      trans
trans      shifttrans
seWithMean TRUEFALSEseWithMean=2TRUE
unconditional TRUE
by.resids  by
scheme
deriv      TRUE
...

```

[termplot](#)

```
deriv=TRUE
```

```
plot.gamrandom.effectsrmfSpherical.Splinefactor.smooth.interaction
```

```
s(cov,edf) covedfscheme == 0scheme == 1scheme == 2se
```

```
scheme==0s(var1,var2,edf)se=TRUEse
```

```
scheme==1scheme==2scheme==3contour.col
```

```
n3vis.gam
```

```
termplotterms
```

```
seWithMean=TRUEseWithMean=2
```

```
imagehcolorsheat.colorsshcolors=rainbow(50)greyhcolors=grey(0:50/50)contour.col
```

```
plot.gamsmooth.constructPredict.matrixfoo.smoothmgcv::plot.mgcv.smooth
```

```
plot.foo.smooths(x,bs="foo")
```



```
plot.gam()
rug=FALSEtoo.far=0
```

```
all.termstermplotgam
```

```
<simon.wood@r-project.org>
<henric.nilsson@statisticon.se>shade
```

```
gampredict.gamvis.gam
```

```
library(mgcv)
set.seed(0)
## fake some data...
f1 <- function(x) {exp(2 * x)}
f2 <- function(x) {
  0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10
}
f3 <- function(x) {x*0}

n<-200
sig2<-4
x0 <- rep(1:4,50)
x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
x3 <- runif(n, 0, 1)
e <- rnorm(n, 0, sqrt(sig2))
y <- 2*x0 + f1(x1) + f2(x2) + f3(x3) + e
x0 <- factor(x0)

## fit and plot...
b<-gam(y~x0+s(x1)+s(x2)+s(x3))
plot(b,pages=1,residuals=TRUE,all.terms=TRUE,shade=TRUE,shade.col=2)
plot(b,pages=1,seWithMean=TRUE) ## better coverage intervals
plot(b,pages=1,scale=0,scheme=2,deriv=TRUE) ## plot derivs of smooths

## just parametric term alone...
```

```

termplot(b, terms="x0", se=TRUE)

## more use of color...
op <- par(mfrow=c(2,2),bg="blue")
x <- 0:1000/1000
for (i in 1:3) {
  plot(b,select=i,rug=FALSE,col="green",
       col.axis="white",col.lab="white",all.terms=TRUE)
  for (j in 1:2) axis(j,col="white",labels=FALSE)
  box(col="white")
  eval(parse(text=paste("fx <- f",i,"(x)",sep="")))
  fx <- fx-mean(fx)
  lines(x,fx,col=2) ## overlay `truth' in red
}
par(op)

## example with 2-d plots, and use of schemes...
b1 <- gam(y~x0+s(x1,x2)+s(x3))
op <- par(mfrow=c(2,2))
plot(b1,all.terms=TRUE)
par(op)
op <- par(mfrow=c(2,2))
plot(b1,all.terms=TRUE,scheme=1)
par(op)
op <- par(mfrow=c(2,2))
plot(b1,all.terms=TRUE,scheme=c(2,1))
par(op)

## 3 and 4 D smooths can also be plotted
dat <- gamSim(1,n=400)
b1 <- gam(y~te(x0,x1,x2,d=c(1,2),k=c(5,15))+s(x3),data=dat)

## Now plot. Use cex.lab and cex.axis to control axis label size,
## n3 to control number of panels, n2 to control panel grid size,
## scheme=1 to get greyscale...

plot(b1,pages=1)

```

polys.plot

```
polys.plot(pc,z=NULL,scheme="heat",lab="",...)
```

pc	NA mr f
z	pczpcpczzNULL
scheme	"heat""grey"z
lab	
...	zNULL

[mrf](#)

<simon.wood@r-project.org>

[mrfcolumb.polys](#)

```
## see also ?mrf for use of z
require(mgcv)
data(columb.polys)
polys.plot(columb.polys)
```

[predict.bam](#)

[predict.gambam](#)discrete=TRUE

[bam](#)[bamR](#)

```
## S3 method for class 'bam'
predict(object,newdata,type="link",se.fit=FALSE,terms=NULL,
        exclude=NULL,block.size=50000,newdata.guaranteed=FALSE,
        na.action=na.pass,cluster=NULL,discrete=TRUE,n.threads=1,gc.level=0,...)
```

object	bam bam
newdata	newdata
type	"link"type="terms"type="iterms"type="response"type="lpmatrix" se.fitR
se.fit	
terms	type=="terms"type="iterms"NULL"(Intercept)"
exclude	type=="terms"type="iterms"NULLnewdata.guaranteed=TRUEnewdata
block.size	
newdata.guaranteed	TRUEnewdatanewdataNA
na.action	NAnewdatana.passnewdataNANANana.excludena.omitnewdataNAnewdata NAobject\$na.action
cluster	predict.bamparallel bam
discrete	TRUEFALSE
n.threads	se.fit=TRUE
gc.level	
...	

```
predict.gamVp
termplot(object"para.only"type=="terms"termplot
bam
predict.gamlpmatrix
discrete=TRUEnewdatabam
```

```
type=="lpmatrix"se.fitTRUEfitse.fitttype"terms"
newdata
```

```
predict.gam
```

```
<simon.wood@r-project.org>
```

```
bampredict.gam
```

```
## for parallel computing see examples for ?bam
## for general useage follow examples in ?predict.gam
```

```
predict.gam
```

```
gamgam()R
```

```
## S3 method for class 'gam'
predict(object,newdata,type="link",se.fit=FALSE,terms=NULL,
        exclude=NULL,block.size=NULL,newdata.guaranteed=FALSE,
        na.action=na.pass,unconditional=FALSE,iterms.type=NULL,...)
```

```

object      gamgam()
newdata     newdatalink{linear.functional.terms}
type        "link"type="terms"type="iterms"type="response"type="lpmatrix"
            se.fitR

se.fit

terms        type=="terms"type="iterms"NULL
exclude      type=="terms"type="iterms"NULLnewdata.guaranteed=TRUEnewdata
block.size   NULL
newdata.guaranteed
            TRUEnewdatanewdataNA
na.action    NAnewdatana.passnewdataNANANAna.excludena.omitnewdataNAnewdata
            NAobject$na.action
unconditional TRUE
iterms.type  type="iterms"iterms.type=2
...

```

```

predict.gamVp
linear.functional.termsnewdatanewdatalinear.functional.terms
termplotobject"para.only"type=="terms"termplot
gam
lpmatrix

```

```

type=="lpmatrix"se.fitTRUEfitse.fitttype"terms"se.fitse.fitllul
newdata

```

```

predict.gam()
type=="terms"predict.lm

```

```

<simon.wood@r-project.org>

```

```

gamgamplot.gam

```

```

library(mgcv)
n <- 200
sig <- 2
dat <- gamSim(1,n=n,scale=sig)

b <- gam(y~s(x0)+s(I(x1^2))+s(x2)+offset(x3),data=dat)

newd <- data.frame(x0=(0:30)/30,x1=(0:30)/30,x2=(0:30)/30,x3=(0:30)/30)
pred <- predict.gam(b,newd)
pred0 <- predict(b,newd,exclude="s(x0)") ## prediction excluding a term
## ...and the same, but without needing to provide x0 prediction data...
newd1 <- newd;newd1$x0 <- NULL ## remove x0 from `newd1`
pred1 <- predict(b,newd1,exclude="s(x0)",newdata.guaranteed=TRUE)

## custom perspective plot...

m1 <- 20;m2 <- 30; n <- m1*m2
x1 <- seq(.2,.8,length=m1);x2 <- seq(.2,.8,length=m2) ## marginal grid points
df <- data.frame(x0=rep(.5,n),x1=rep(x1,m2),x2=rep(x2,each=m1),x3=rep(0,n))
pf <- predict(b,newdata=df,type="terms")
persp(x1,x2,matrix(pf[,2]+pf[,3],m1,m2),theta=-130,col="blue",zlab="")

#####
## difference between "terms" and "iterms"
#####
nd2 <- data.frame(x0=c(.25,.5),x1=c(.25,.5),x2=c(.25,.5),x3=c(.25,.5))
predict(b,nd2,type="terms",se=TRUE)
predict(b,nd2,type="iterms",se=TRUE)

#####
## now get variance of sum of predictions using lpmatrix
#####

Xp <- predict(b,newd,type="lpmatrix")

## Xp %*% coef(b) yields vector of predictions

a <- rep(1,31)
Xs <- t(a) %*% Xp ## Xs %*% coef(b) gives sum of predictions
var.sum <- Xs %*% b$Vp %*% t(Xs)

#####
## Now get the variance of non-linear function of predictions
## by simulation from posterior distribution of the params
#####

rmvn <- function(n,mu,sig) { ## MVN random deviates
  L <- mroot(sig);m <- ncol(L);
  t(mu + L%*%matrix(rnorm(m*n),m,n))
}

br <- rmvn(1000,coef(b),b$Vp) ## 1000 replicate param. vectors
res <- rep(0,1000)
for (i in 1:1000)
{ pr <- Xp %*% br[i,] ## replicate predictions

```

```

    res[i] <- sum(log(abs(pr))) ## example non-linear function
  }
  mean(res);var(res)

## loop is replace-able by following ....

res <- colSums(log(abs(Xp %*% t(br))))

#####
## The following shows how to use an "lpmatrix" as a lookup
## table for approximate prediction. The idea is to create
## approximate prediction matrix rows by appropriate linear
## interpolation of an existing prediction matrix. The additivity
## of a GAM makes this possible.
## There is no reason to ever do this in R, but the following
## code provides a useful template for predicting from a fitted
## gam *outside* R: all that is needed is the coefficient vector
## and the prediction matrix. Use larger `Xp`/ smaller `dx` and/or
## higher order interpolation for higher accuracy.
#####

xn <- c(.341,.122,.476,.981) ## want prediction at these values
x0 <- 1      ## intercept column
dx <- 1/30    ## covariate spacing in `newd`
for (j in 0:2) { ## loop through smooth terms
  cols <- 1+j*9 +1:9      ## relevant cols of Xp
  i <- floor(xn[j+1]*30)  ## find relevant rows of Xp
  w1 <- (xn[j+1]-i*dx)/dx ## interpolation weights
  ## find approx. predict matrix row portion, by interpolation
  x0 <- c(x0,Xp[i+2,cols]*w1 + Xp[i+1,cols]*(1-w1))
}
dim(x0)<-c(1,28)
fv <- x0%*%coef(b) + xn[4];fv      ## evaluate and add offset
se <- sqrt(x0%*%b$Vp%*%t(x0));se ## get standard error
## compare to normal prediction
predict(b,newdata=data.frame(x0=xn[1],x1=xn[2],
                             x2=xn[3],x3=xn[4]),se=TRUE)

#####
## Example of producing a prediction interval for non Gaussian
## data...
#####

f <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10
set.seed(6);n <- 100;x <- sort(runif(n))
Ey <- exp(f(x)/4);scale <- .5
y <- rgamma(n,shape=1/scale,scale=Ey*scale) ## sim gamma dataexit
b <- gam(y~s(x,k=20),family=Gamma(link=log),method="REML")
Xp <- predict(b,type="lpmatrix")
br <- rmvn(10000,coef(b),vcov(b)) ## 1000 replicate param. vectors
fr <- Xp
yr <- apply(fr,2,function(x) rgamma(length(x),shape=1/b$scale,
  scale=exp(x)*b$scale)) ## replicate data
pi <- apply(yr,1,quantile,probs=c(.1,.9),type=9) ## 80% PI
plot(x,y);lines(x,fitted(b));lines(x,pi[1,]);lines(x,pi[2,])

```

```

mean(y>pi[1,]&y<pi[2,]) ## check it

#####
# illustration of unsafe scale dependent transforms in smooths...
#####

b0 <- gam(y~s(x0)+s(x1)+s(x2)+x3,data=dat) ## safe
b1 <- gam(y~s(x0)+s(I(x1/2))+s(x2)+scale(x3),data=dat) ## safe
b2 <- gam(y~s(x0)+s(scale(x1))+s(x2)+scale(x3),data=dat) ## unsafe
pd <- dat; pd$x1 <- pd$x1/2; pd$x3 <- pd$x3/2
par(mfrow=c(1,2))
plot(predict(b0,pd),predict(b1,pd),main="b0 and b1 predictions match")
abline(0,1,col=2)
plot(predict(b0,pd),predict(b2,pd),main="b2 unsafe, doesn't match")
abline(0,1,col=2)

#####
## Differentiating the smooths in a model (with CIs for derivatives)
#####

## simulate data and fit model...
dat <- gamSim(1,n=300,scale=sig)
b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b,pages=1)

## now evaluate derivatives of smooths with associated standard
## errors, by finite differencing...
x.mesh <- seq(0,1,length=200) ## where to evaluate derivatives
newd <- data.frame(x0 = x.mesh,x1 = x.mesh, x2=x.mesh,x3=x.mesh)
X0 <- predict(b,newd,type="lpmatrix")

eps <- 1e-7 ## finite difference interval
x.mesh <- x.mesh + eps ## shift the evaluation mesh
newd <- data.frame(x0 = x.mesh,x1 = x.mesh, x2=x.mesh,x3=x.mesh)
X1 <- predict(b,newd,type="lpmatrix")

Xp <- (X1-X0)/eps ## maps coefficients to (fd approx.) derivatives
colnames(Xp)      ## can check which cols relate to which smooth

par(mfrow=c(2,2))
for (i in 1:4) { ## plot derivatives and corresponding CIs
  Xi <- Xp*0
  Xi[(i-1)*9+1:9+1] <- Xp[(i-1)*9+1:9+1] ## Xi%*%coef(b) = smooth deriv i
  df <- Xi%*%coef(b) ## ith smooth derivative
  df.sd <- rowSums(Xi%*%b$Vp*Xi)^.5 ## cheap diag(Xi%*%b$Vp%*%t(Xi))^0.5
  plot(x.mesh,df,type="l",ylim=range(c(df+2*df.sd,df-2*df.sd)))
  lines(x.mesh,df+2*df.sd,lty=2);lines(x.mesh,df-2*df.sd,lty=2)
}

```


smoothsmooth.construct

[PredictMat](#)

Predict.matrix(object,data)

Predict.matrix2(object,data)

object smooth.constructPredict.matrix[smooth.construct](#)

data [smooth.construct](#)smooth.construct2

xx.smooth.specsmooth.constructPredict.matrix

smooth.construct[Predict.matrixsmooth.construct](#)

object"offset"

<simon.wood@r-project.org>

[gamgamsmooth.constructPredictMat](#)

See smooth.construct examples

Predict.matrix.cr.smooth

[gamPredict.matrix](#)

```
## S3 method for class 'cr.smooth'
Predict.matrix(object, data)
## S3 method for class 'cs.smooth'
Predict.matrix(object, data)
## S3 method for class 'cyclic.smooth'
Predict.matrix(object, data)
## S3 method for class 'pspline.smooth'
Predict.matrix(object, data)
## S3 method for class 'tensor.smooth'
Predict.matrix(object, data)
## S3 method for class 'tprs.smooth'
Predict.matrix(object, data)
## S3 method for class 'ts.smooth'
Predict.matrix(object, data)
## S3 method for class 't2.smooth'
Predict.matrix(object, data)
```

```
object      smooth.constructstegam
data        smooth.constructsmooth.construct2
```

```
predict.gamgamPredict.matrixsmooth.construct
```

```
<simon.wood@r-project.org>
```

```
## see smooth.construct
```

```
Predict.matrix.soap.film
```

```
Predict.matrixgam
```

```
## S3 method for class 'soap.film'
Predict.matrix(object,data)
## S3 method for class 'sf.wiggly'
Predict.matrix(object,data)
## S3 method for class 'sf.film'
Predict.matrix(object,data)
```

```
object      "soap.film""sf.wiggly""sf.film"
data
```

```
smooth.construct.so.smooth.specXS
```

```
"offset"
```

```
smooth.construct.so.smooth.spec
```

```
## This is a lower level example. The basis and
## penalties are obtained explicitly
## and 'magic' is used as the fitting routine...
```

```
require(mgcv)
set.seed(66)
```

```
## create a boundary...
fsb <- list(fs.boundary())
```

```
## create some internal knots...
knots <- data.frame(x=rep(seq(-.5,3,by=.5),4),
                    y=rep(c(-.6,-.3,.3,.6),rep(8,4)))
```

```
## Simulate some fitting data, inside boundary...
n<-1000
```

```
x <- runif(n)*5-1;y<-runif(n)*2-1
z <- fs.test(x,y,b=1)
ind <- inSide(fsb,x,y) ## remove outsiders
z <- z[ind];x <- x[ind]; y <- y[ind]
n <- length(z)
z <- z + rnorm(n)*.3 ## add noise
```

```
## plot boundary with knot and data locations
plot(fsb[[1]]$x,fsb[[1]]$y,type="l");points(knots$x,knots$y,pch=20,col=2)
points(x,y,pch=".",col=3);
```

```
## set up the basis and penalties...
sob <- smooth.construct2(s(x,y,bs="so",k=40,xt=list(bnd=fsb,nmax=100)),
                        data=data.frame(x=x,y=y),knots=knots)
## ... model matrix is element 'X' of sob, penalties matrices
```

```

## are in list element `S'.

## fit using `magic'
um <- magic(z,sob$X,sp=c(-1,-1),sob$S,off=c(1,1))
beta <- um$b

## produce plots...
par(mfrow=c(2,2),mar=c(4,4,1,1))
m<-100;n<-50
xm <- seq(-1,3.5,length=m);yn<-seq(-1,1,length=n)
xx <- rep(xm,n);yy<-rep(yn,rep(m,n))

## plot truth...
tru <- matrix(fs.test(xx,yy),m,n) ## truth
image(xm,yn,tru,col=heat.colors(100),xlab="x",ylab="y")
lines(fsb[[1]]$x,fsb[[1]]$y,lwd=3)
contour(xm,yn,tru,levels=seq(-5,5,by=.25),add=TRUE)

## Plot soap, by first predicting on a fine grid...

## First get prediction matrix...
X <- Predict.matrix2(sob,data=list(x=xx,y=yy))

## Now the predictions...
fv <- X*%beta

## Plot the estimated function...
image(xm,yn,matrix(fv,m,n),col=heat.colors(100),xlab="x",ylab="y")
lines(fsb[[1]]$x,fsb[[1]]$y,lwd=3)
points(x,y,pch=".")
contour(xm,yn,matrix(fv,m,n),levels=seq(-5,5,by=.25),add=TRUE)

## Plot TPRS...
b <- gam(z~s(x,y,k=100))
fv.gam <- predict(b,newdata=data.frame(x=xx,y=yy))
names(sob$sd$bnd[[1]]) <- c("xx","yy","d")
ind <- inSide(sob$sd$bnd,xx,yy)
fv.gam[!ind]<-NA
image(xm,yn,matrix(fv.gam,m,n),col=heat.colors(100),xlab="x",ylab="y")
lines(fsb[[1]]$x,fsb[[1]]$y,lwd=3)
points(x,y,pch=".")
contour(xm,yn,matrix(fv.gam,m,n),levels=seq(-5,5,by=.25),add=TRUE)

```

```
print.gam
```

```
gam
```

```
## S3 method for class 'gam'
print(x, ...)
```

```
x...      gamgam()
```

`gamObjectnames(x)summary.gam`

`gamm`

`<simon.wood@r-project.org>`

`gamsummary.gam`

`psum.chisq`

$$P(q < \sum_{i=1}^r \lambda_i X_i + \sigma_z Z)$$

`X_jdf[j]nc[j]Z`

`psum.chisq(q,lb,df=rep(1,length(lb)),nc=rep(0,length(lb)),sigz=0,
lower.tail=FALSE,tol=2e-5,nlim=100000,trace=FALSE)`

`q
lb λii
df
nc
sigz
lower.tail
tol
nlim
trace TRUE`

`gotolblog(1+x)qpsum.chisqq[i]
NA
traceTRUE"ifault""trace"qq`

`<simon.wood@r-project.org>`

```

require(mgcv)
lb <- c(4.1,1.2,1e-3,-1) ## weights
df <- c(2,1,1,1) ## degrees of freedom
nc <- c(1,1.5,4,1) ## non-centrality parameter
q <- c(1,6,20) ## quantiles to evaluate

psum.chisq(q,lb,df,nc)

## same by simulation...

psc.sim <- function(q,lb,df=lb*0+1,nc=df*0,ns=10000) {
  r <- length(lb); p <- q
  X <- rowSums(rep(lb,each=ns) *
    matrix(rchisq(r*ns,rep(df,each=ns),rep(nc,each=ns)),ns,r))
  apply(matrix(q),1,function(q) mean(X>q))
} ## psc.sim

psum.chisq(q,lb,df,nc)
psc.sim(q,lb,df,nc,100000)

```

qq.gam

gamgam()

```

qq.gam(object, rep=0, level=.9,s.rep=10,
        type=c("deviance","pearson","response"),
        pch=".", rl.col=2, rep.col="gray80", ...)

```

```

object      gamgam()glm
rep          0
level
s.rep
type        residuals.gam
pch
rl.col
rep.col
...

```

```
s.rep  
rep  
rep
```

<simon.wood@r-project.org>

[choose.kgam](#)

```
library(mgcv)  
## simulate binomial data...  
set.seed(0)  
n.samp <- 400  
dat <- gamSim(1,n=n.samp,dist="binary",scale=.33)  
p <- binomial()$linkinv(dat$f) ## binomial p  
n <- sample(c(1,3),n.samp,replace=TRUE) ## binomial n  
dat$y <- rbinom(n,n,p)  
dat$n <- n  
  
lr.fit <- gam(y~s(x0)+s(x1)+s(x2)+s(x3)  
             ,family=binomial,data=dat,weights=n,method="REML")  
  
par(mfrow=c(2,2))  
## normal QQ-plot of deviance residuals  
qqnorm(residuals(lr.fit),pch=19,cex=.3)  
## Quick QQ-plot of deviance residuals  
qq.gam(lr.fit,pch=19,cex=.3)  
## Simulation based QQ-plot with reference bands  
qq.gam(lr.fit,rep=100,level=.9)  
## Simulation based QQ-plot, Pearson resids, all  
## simulated reference plots shown...  
qq.gam(lr.fit,rep=100,level=1,type="pearson",pch=19,cex=.2)  
  
## Now fit the wrong model and check....  
  
pif <- gam(y~s(x0)+s(x1)+s(x2)+s(x3)  
           ,family=poisson,data=dat,method="REML")  
par(mfrow=c(2,2))  
qqnorm(residuals(pif),pch=19,cex=.3)  
qq.gam(pif,pch=19,cex=.3)  
qq.gam(pif,rep=100,level=.9)  
qq.gam(pif,rep=100,level=1,type="pearson",pch=19,cex=.2)
```

```
## Example of binary data model violation so gross that you see a problem
## on the QQ plot...

y <- c(rep(1,10),rep(0,20),rep(1,40),rep(0,10),rep(1,40),rep(0,40))
x <- 1:160
b <- glm(y~x,family=binomial)
par(mfrow=c(2,2))
## Note that the next two are not necessarily similar under gross
## model violation...
qq.gam(b)
qq.gam(b,rep=50,level=1)
## and a much better plot for detecting the problem
plot(x,residuals(b),pch=19,cex=.3)
plot(x,y);lines(x,fitted(b))

## alternative model
b <- gam(y~s(x,k=5),family=binomial,method="ML")
qq.gam(b)
qq.gam(b,rep=50,level=1)
plot(x,residuals(b),pch=19,cex=.3)
plot(b,residuals=TRUE,pch=19,cex=.3)
```

random.effects

```
gammmamm4gamm4
gams(...,bs="re")smooth.construct.re.smooth.specs(x,z,g,bs="re")
model.matrix(~x:z:g-1)gs(g,bs="re")ggxs(x,g,bs="re")xghs(h,g,bs="re")ghxts
linear.functional.terms
paraPengamm.models
gam(...,method="REML")
gammammamm4
gamm.vcomp
summary.gamanova.gam
```

```
gam.vcompgam.modelssmooth.termssmooth.construct.re.smooth.specgamm
```



```

## see also examples for gam.models, gam.vcomp, gamm
## and smooth.construct.re.smooth.spec

## simple comparison of lme and gam
require(mgcv)
require(nlme)
b0 <- lme(travel~1,data=Rail,~1|Rail,method="REML")

b <- gam(travel~s(Rail,bs="re"),data=Rail,method="REML")

intervals(b0)
gam.vcomp(b)
anova(b)
plot(b)

## simulate example...
dat <- gamSim(1,n=400,scale=2) ## simulate 4 term additive truth

fac <- sample(1:20,400,replace=TRUE)
b <- rnorm(20)*.5
dat$y <- dat$y + b[fac]
dat$fac <- as.factor(fac)

rm1 <- gam(y ~ s(fac,bs="re")+s(x0)+s(x1)+s(x2)+s(x3),data=dat,method="ML")
gam.vcomp(rm1)

fv0 <- predict(rm1,exclude="s(fac)") ## predictions setting r.e. to 0
fv1 <- predict(rm1) ## predictions setting r.e. to predicted values
## prediction setting r.e. to 0 and not having to provide 'fac'...
pd <- dat; pd$fac <- NULL
fv0 <- predict(rm1,pd,exclude="s(fac)",newdata.guaranteed=TRUE)

## Prediction with levels of fac not in fit data.
## The effect of the new factor levels (or any interaction involving them)
## is set to zero.
xx <- seq(0,1,length=10)
pd <- data.frame(x0=xx,x1=xx,x2=xx,x3=xx,fac=c(1:10,21:30))
fv <- predict(rm1,pd)
pd$fac <- NULL
fv0 <- predict(rm1,pd,exclude="s(fac)",newdata.guaranteed=TRUE)

```

residuals.gam

gam

```

## S3 method for class 'gam'
residuals(object, type = "deviance",...)

```

```
object      gam
type        "deviance""pearson""scaled.pearson""working""response"
...
```

$(y - \mu) / \sqrt{V(\mu)}$
[cox.ph](#)

<simon.wood@r-project.org>

[gam](#)

[rig](#)

`rig(n,mean,scale)`

n
mean
scale

`meanscale*mean^3statmodscale`

<simon.wood@r-project.org>

```

require(mgcv)
set.seed(7)
## An inverse.gaussian GAM example, by modify `gamSim' output...
dat <- gamSim(1,n=400,dist="normal",scale=1)
dat$f <- dat$f/4 ## true linear predictor
Ey <- exp(dat$f);scale <- .5 ## mean and GLM scale parameter
## simulate inverse Gaussian response...
dat$y <- rig(Ey,mean=Ey,scale=.2)
big <- gam(y~ s(x0)+ s(x1)+s(x2)+s(x3),family=inverse.gaussian(link=log),
          data=dat,method="REML")
plot(big,pages=1)
gam.check(big)
summary(big)

```

rmvn

```

rmvn(n,mu,V)
r.mvt(n,mu,V,df)
dmvn(x,mu,V,R=NULL)
d.mvt(x,mu,V,df,R=NULL)

```

```

n
mu          p=ncol(V)np
V
df
x
R

```

V

nVmumu

<simon.wood@r-project.org>

[ldTweedieTweedie](#)

```
library(mgcv)
V <- matrix(c(2,1,1,2),2,2)
mu <- c(1,3)
n <- 1000
z <- rmvn(n,mu,V)
crossprod(sweep(z,2,colMeans(z)))/n ## observed covariance matrix
colMeans(z) ## observed mu
dmvn(z,mu,V)
```

Rrank

rankrankrank

Rrank(R,tol=.Machine\$double.eps^.9)

R

tol

<simon.wood@r-project.org>

```
set.seed(0)
n <- 10;p <- 5
x <- runif(n*(p-1))
X <- matrix(c(x,x[1:n]),n,p)
qrx <- qr(X,LAPACK=TRUE)
Rrank(qr.R(qrx))
```

rTweedie

rTweedie(mu,p=1.5,phi=1)

mu	mu
p	mupphi=1
phi	phi*mu^p

$\frac{\mu^{\frac{1}{p-1}}}{\Gamma(\frac{1}{p-1})} \frac{\phi^{\frac{1}{p-1}}}{\Gamma(\frac{1}{p-1})} \mu^{\frac{1}{p-1}} \phi^{\frac{1}{p-1}}$
rtweedietweedie

$\mu \phi^{\frac{1}{p-1}}$

<simon.wood@r-project.org>

<https://cran.r-project.org/package=tweedie>

ldTweedieTweedie

```
library(mgcv)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10
n <- 300
x <- runif(n)
mu <- exp(f2(x)/3+.1); x <- x*10 - 4
y <- rTweedie(mu,p=1.5,phi=1.3)
b <- gam(y~s(x,k=20),family=Tweedie(p=1.5))
b
plot(b)
```

s

gam

s(..., k=-1, fx=FALSE, bs="tp", m=NA, by=NA, xt=NULL, id=NULL, sp=NULL, pc=NULL)

...	s(log(x))s(I(x/sd(x))) predict.gam
k	knull.space.dimensionchoose.k
fx	TRUEFALSE
bs	"tp""cr" smooth.terms
m	NA"ps"
by	byby variableby gam.models by gam.models linear.functional.terms
xt	"tp"xt\$sumConvFALSE
id	ididby
sp	sp gam gamm
pc	NULL identi fiability scasm

[smooth.construct](#)

xx.smooth.specxxbsssmooth.specsmooth.construct

term	
bs.dim	
fixed	
dim	
p.order	
by	by"NA"
label	
xt	xt
id	NULL
sp	NULL

<simon.wood@r-project.org>

tegamgamm

```
# example utilising `by' variables
library(mgcv)
set.seed(0)
n<-200;sig2<-4
x1 <- runif(n, 0, 1);x2 <- runif(n, 0, 1);x3 <- runif(n, 0, 1)
fac<-c(rep(1,n/2),rep(2,n/2)) # create factor
fac.1<-rep(0,n)+(fac==1);fac.2<-1-fac.1 # and dummy variables
fac<-as.factor(fac)
f1 <- exp(2 * x1) - 3.75887
f2 <- 0.2 * x1^11 * (10 * (1 - x1))^6 + 10 * (10 * x1)^3 * (1 - x1)^10
f<-f1*fac.1+f2*fac.2+x2
e <- rnorm(n, 0, sqrt(abs(sig2)))
y <- f + e
# NOTE: smooths will be centered, so need to include fac in model....
b<-gam(y~fac+s(x1,by=fac)+x2)
plot(b,pages=1)
```

scasm

family.mgcv

```
scasm(formula,family=gaussian(),data=list(),weights=NULL,subset=NULL,
      na.action, offset=NULL,control=list(),scale=0,select=FALSE,
      knots=NULL,sp=NULL,gamma=1,fit=TRUE,G=NULL,drop.unused.levels=TRUE,
      drop.intercept=NULL,bs=0,...)
```

formula	formula.gamgam.modelsstetit2
family	glmfamilyfamily.mgcvquasimethod
data	environment(formula)gam
weights	weights <- weights/mean(weights)
subset	
na.action	
offset	formula mg lm
control	gam.control
select	gamma
scale	
gamma	gamma*log(n)/2n/gamma
knots	kktprs "tp"/"ts"
sp	length(sp)discrete=TRUEsp

drop.unused.levels

```
G          NULLbamfit=FALSEspchunk.sizegammanthreadsclusterrhogc.level
          samfracuse.cholmethodscale
fit         FALSEGbam
drop.intercept TRUE
bs          FALSEbs
...         gam.fitmustart
```

gam

[smooth.construct.sc.smooth.specpcspc](#)

$f(X, Z)\mathbf{XZ}\mathbf{W}\mathbf{b}$

$$\sum_j f(X_{ij}, Z_{ij})W_{ij} > b_i$$

b

[smooth.construct.sc.smooth.specpc](#)

"gam"gamObject

<simon.wood@r-project.org>

[smooth.construct.sc.smooth.specmgcv-packagegamObjectgam.modelssmooth.terms](#)
[predict.gamplot.gamsummary.gamgam.check](#)

```
library(mgcv)
## a density estimation example, illustrating general constraint use...
n <- 400; set.seed(4)
x <- c(rnorm(n*.8,.4,.13),runif(n*.2))
y <- rep(1e-10,n)
xp=(1:n-.5)/n
options(warn=-1)
## Use an exponential distribution, inverse link and zero pseudoresponse
## data to get correct likelihood. Use a positive smooth and impose
## integration to 1 via a general linear 'pc' constraint...
b <- scasm(y ~ s(x,bs="sc",xt="+",k=15,pc=list(list(x=matrix(xp,1,n),
  matrix(1/n,1,n),1))),-1, family=Gamma,scale=1,knots=list(x=c(0,1)))
options(warn=0)
plot(b,scheme=2)
```



```

lines(xp,.2+dnorm(xp,.4,.13)*.8,col=2)

## some more standard examples

fs <- function(x,k) { ## some simulation functions
  if (k==2) 0.2*x^11*(10*(1-x))^6 + 10*(10*x)^3*(1-x)^10 else
  if (k==0) 2 * sin(pi * x) else
  if (k==1) exp(2 * x) else
  if (k==3) exp((x-.4)*20)/(1+exp((x-.4)*20)) else
  if (k==4) (x-.55)^4 else x
} ## fs

## Simple Poisson example...
n <- 400; set.seed(4)
x0 <- runif(n); x1 <- runif(n); x2 <- runif(n); x3 <- runif(n);
fac <- factor(sample(1:40,n,replace=TRUE))
f <- fs(x0,k=0) + 2*fs(x1,k=3) + fs(x2,k=2) + 10*fs(x3,k=4)
mu <- exp((f-1)/4)
y <- rpois(n,mu)
k <- 10
b0 <- gam(y~s(x0,bs="bs",k=k)+s(x1,bs="bs",k=k)+s(x2,bs="bs",k=k)+
  s(x3,bs="bs",k=k),method="REML",family=poisson)

b1 <- scasm(y~s(x0,bs="bs",k=k)+s(x1,bs="bs",k=k)+s(x2,bs="bs",k=k)+
  s(x3,bs="bs",k=k),family=poisson)
b0;b1 ## note similarity when no constraints

plot(b1,pages=1,scheme=2) ## second term not monotonic, so impose this...

b2 <- scasm(y~s(x0,bs="bs",k=k)+s(x1,bs="sc",xt="m+",k=k)+
  s(x2,bs="bs",k=k)+s(x3,bs="bs",k=k),family=poisson)
plot(b2,pages=1,scheme=2)

## constraint respecting bootstrap CIs...

b3 <- scasm(y~s(x0,bs="bs",k=k)+s(x1,bs="sc",xt="m+",k=k)+
  s(x2,bs="bs",k=k)+s(x3,bs="bs",k=k),family=poisson,bs=200)
plot(b3,pages=1,scheme=2)
fv <- predict(b3,se=TRUE)
fv1 <- predict(b3,se=.95)
plot(fv$se*3.92,fv1$ul-fv1$ll) ## compare CI widths
abline(0,1,col=2)

## gamma distribution location-scale example

## simulate data...
x0 <- runif(n); x1 <- runif(n); x2 <- runif(n); x3 <- runif(n);
fac <- factor(sample(1:40,n,replace=TRUE))
f <- fs(x0,k=0) + 2*fs(x1,k=3) + fs(x2,k=2) #+ 10*fs(x3,k=4)
mu <- exp((fs(x0,k=0)+ fs(x2,k=2))/5)
th <- exp((fs(x1,k=1)+ 10*fs(x3,k=4))/2-2)
y <- rgamma(n,shape=1/th,scale=mu*th)

## fit model
b <- scasm(list(y~s(x0,bs="sc",xt="c-")+s(x2),~s(x1,bs="sc",xt="m+")
  +s(x3)),family=gammals,bs=200)
plot(b,scheme=2,pages=1,scale=0) ## bootstrap CIs

```

```

b$bs <- NULL; plot(b,scheme=2,pages=1,scale=0) ## constraint free CI

## A survival example...

library(survival) ## for data
col1 <- colon[colon$type==1,] ## focus on single event
col1$differ <- as.factor(col1$differ)
col1$sex <- as.factor(col1$sex)

## set up the AFT response...
logt <- cbind(log(col1$time),log(col1$time))
logt[col1$status==0,2] <- Inf ## right censoring
col1$logt <- -logt ## -ve conventional for AFT versus Cox PH comparison

## fit the model...
b0 <- gam(logt~s(age,by=sex)+sex+s(nodes)+perfor+rx+obstruct+adhere,
          family=cnorm(),data=col1)
plot(b0,pages=1)

## nodes effect should be increasing...
b <- scasm(logt~s(age,by=sex)+sex+s(nodes,bs="sc",xt="m+")+perfor+rx+
          obstruct+adhere,family=cnorm(),data=col1)
plot(b,pages=1)
b

## same with logistic distribution...
b <- scasm(logt~s(age,by=sex)+sex+s(nodes,bs="sc",xt="m+")+perfor+rx+
          obstruct+adhere,family=clog(),data=col1)
plot(b,pages=1)

## and with Cox PH model...
b <- scasm(time~s(age,by=sex)+sex+s(nodes,bs="sc",xt="m+")+perfor+rx+
          obstruct+adhere,family=cox.ph(),data=col1,weights=status)
summary(b)
plot(b,pages=1) ## plot effects

```

scat

$\text{gambam}(y - \mu)/\sigma \sim t_{\nu, \mu\sigma\nu}$

scat(theta = NULL, link = "identity",min.df=3)

theta $\nu = b + \exp(\theta_1)\text{min.df}$ $\sigma = \exp(\theta_2)\nu\sigma$

link "identity""log""inverse"

min.df

min.dfmin.df

extended.family

```
library(mgcv)
## Simulate some t data...
set.seed(3);n<-400
dat <- gamSim(1,n=n)
dat$y <- dat$f + rt(n,df=4)*2

b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=scat(link="identity"),data=dat)

b
plot(b,pages=1)
```

sdiag

```
sdiag(A,k=0)
sdiag(A,k=0) <- value
```

A
k
value

value

<simon.wood@r-project.org>

```
require(mgcv)
A <- matrix(1:35,7,5)
A
sdiag(A,1) ## first super diagonal
sdiag(A,-1) ## first sub diagonal

sdiag(A) <- 1 ## leading diagonal set to 1
sdiag(A,3) <- c(-1,-2) ## set 3rd super diagonal
```

shash

shash

```
shash(link = list("identity", "logeb", "identity", "identity"),
      b = 1e-2, phiPen = 1e-3)
```

link

b

phiPen

$$p(y|\mu, \sigma, \epsilon, \delta) = C(z) \exp\{-S(z)^2/2\} \{2\pi(1+z^2)\}^{-1/2}/\sigma,$$
$$C(z) = \{1 + S(z)^2\}^{1/2} S(z) = \sinh\{\delta \sinh^{-1}(z) - \epsilon\} z = (y - \mu)/(\sigma\delta) \mu\sigma > 0 \epsilon\delta > 0$$
$$\delta > 10 < \delta < 1 \tau = \log(\sigma) \phi = \log(\delta)$$
$$\tau\eta = \log\{\exp(\tau) - b\} \tau = \log(\sigma) = \log\{\exp(\eta) + b\} \sigma - \phi Pen * \phi^2 \phi \phi$$

general.family

```
#####
# Shash dataset
#####
## Simulate some data from shash
set.seed(847)
n <- 1000
x <- seq(-4, 4, length.out = n)

X <- cbind(1, x, x^2)
beta <- c(4, 1, 1)
mu <- X %*% beta

sigma = .5+0.4*(x+4)*.5      # Scale
eps = 2*sin(x)               # Skewness
del = 1 + 0.2*cos(3*x)       # Kurtosis
```

```

dat <- mu + (del*sigma)*sinh((1/del)*asinh(qnorm(runif(n)))) + (eps/del))
dataf <- data.frame(cbind(dat, x))
names(dataf) <- c("y", "x")
plot(x, dat, xlab = "x", ylab = "y")

## Fit model
fit <- gam(list(y ~ s(x), # <- model for location
               ~ s(x),   # <- model for log-scale
               ~ s(x),   # <- model for skewness
               ~ s(x, k = 20)), # <- model for log-kurtosis
            data = dataf,
            family = shash, # <- new family
            optimizer = "efs")

## Plotting truth and estimates for each parameters of the density
muE <- fit$fitted[ , 1]
sigE <- exp(fit$fitted[ , 2])
epsE <- fit$fitted[ , 3]
delE <- exp(fit$fitted[ , 4])

par(mfrow = c(2, 2))
plot(x, muE, type = 'l', ylab = expression(mu(x)), lwd = 2)
lines(x, mu, col = 2, lty = 2, lwd = 2)
legend("top", c("estimated", "truth"), col = 1:2, lty = 1:2, lwd = 2)

plot(x, sigE, type = 'l', ylab = expression(sigma(x)), lwd = 2)
lines(x, sigma, col = 2, lty = 2, lwd = 2)

plot(x, epsE, type = 'l', ylab = expression(epsilon(x)), lwd = 2)
lines(x, eps, col = 2, lty = 2, lwd = 2)

plot(x, delE, type = 'l', ylab = expression(delta(x)), lwd = 2)
lines(x, del, col = 2, lty = 2, lwd = 2)

## Plotting true and estimated conditional density
par(mfrow = c(1, 1))
plot(x, dat, pch = '.', col = "grey", ylab = "y", ylim = c(-35, 70))
for(qq in c(0.001, 0.01, 0.1, 0.5, 0.9, 0.99, 0.999)){
  est <- fit$family$qf(p=qq, mu = fit$fitted)
  true <- mu + (del * sigma) * sinh((1/del) * asinh(qnorm(qq)) + (eps/del))
  lines(x, est, type = 'l', col = 1, lwd = 2)
  lines(x, true, type = 'l', col = 2, lwd = 2, lty = 2)
}
legend("topleft", c("estimated", "truth"), col = 1:2, lty = 1:2, lwd = 2)

#####
## Motorcycle example
#####

# Here shash is overkill, in fact the fit is not good, relative
# to what we would get with mgcv::gauss
library(MASS)

b <- gam(list(accel~s(times, k=20, bs = "ad"), ~s(times, k = 10), ~1, ~1),
            data=mcycle, family=shash)

```

```

par(mfrow = c(1, 1))
xSeq <- data.frame(cbind("accel" = rep(0, 1e3),
                        "times" = seq(2, 58, length.out = 1e3)))
pred <- predict(b, newdata = xSeq)
plot(mcycle$times, mcycle$accel, ylim = c(-180, 100))
for(qq in c(0.1, 0.3, 0.5, 0.7, 0.9)){
  est <- b$family$qf(p=qq, mu = pred)
  lines(xSeq$times, est, type = 'l', col = 2)
}

plot(b, pages = 1, scale = FALSE)

```

single.index

$s(X\alpha)\alpha\|\alpha\| = 1$
 α

<simon.wood@r-project.org>

```

require(mgcv)

si <- function(theta,y,x,z,opt=TRUE,k=10,fx=FALSE) {
  ## Fit single index model using gam call, given theta (defines alpha).
  ## Return ML if opt==TRUE and fitted gam with theta added otherwise.
  ## Suitable for calling from 'optim' to find optimal theta/alpha.
  alpha <- c(1,theta) ## constrained alpha defined using free theta
  kk <- sqrt(sum(alpha^2))
  alpha <- alpha/kk ## so now ||alpha||=1
  a <- x%*%alpha ## argument of smooth
  b <- gam(y~s(a,fx=fx,k=k)+s(z),family=poisson,method="ML") ## fit model
  if (opt) return(b$gcv.ubre) else {
    b$alpha <- alpha ## add alpha
    J <- outer(alpha,-theta/kk^2) ## compute Jacobian
    for (j in 1:length(theta)) J[j+1,j] <- J[j+1,j] + 1/kk
    b$J <- J ## dalpha_i/dtheta_j
    return(b)
  }
} ## si

## simulate some data from a single index model...

set.seed(1)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
  (10 * x)^3 * (1 - x)^10
n <- 200;m <- 3
x <- matrix(runif(n*m),n,m) ## the covariates for the single index part
z <- runif(n) ## another covariate
alpha <- c(1,-1,.5); alpha <- alpha/sqrt(sum(alpha^2))
eta <- as.numeric(f2((x%*%alpha+.41)/1.4)+1+z^2*2)/4
mu <- exp(eta)

```

```

y <- rpois(n,mu) ## Poi response

## now fit to the simulated data...

th0 <- c(-.8,.4) ## close to truth for speed
## get initial theta, using no penalization...
f0 <- nlm(si,th0,y=y,x=x,z=z,fx=TRUE,k=5)
## now get theta/alpha with smoothing parameter selection...
f1 <- nlm(si,f0$estimate,y=y,x=x,z=z,hessian=TRUE,k=10)
theta.est <-f1$estimate

## Alternative using 'optim'...

th0 <- rep(0,m-1)
## get initial theta, using no penalization...
f0 <- optim(th0,si,y=y,x=x,z=z,fx=TRUE,k=5)
## now get theta/alpha with smoothing parameter selection...
f1 <- optim(f0$par,si,y=y,x=x,z=z,hessian=TRUE,k=10)
theta.est <-f1$par

## extract and examine fitted model...

b <- si(theta.est,y,x,z,opt=FALSE) ## extract best fit model
plot(b,pages=1)
b
b$alpha
## get sd for alpha...
Vt <- b$J%%solve(f1$hessian,t(b$J))
diag(Vt)^.5

```

Sl.inirep

inverse==TRUE

Sl.inirep(Sl,X,l,r,nt=1)

Sl.initial.repara(Sl, X, inverse = FALSE, both.sides = TRUE, cov = TRUE,
nt = 1)

Sl	Sl.setup
X	
l	
r	
inverse	TRUE
both.sides	inverse==TRUEboth.sides==FALSEboth.sides==FALSEXinverse==FALSEX
cov	X
nt	

X

S1.repara

ldetSX

S1.repara(rp, X, inverse = FALSE, both.sides = TRUE)

rp

X XX

inverse TRUE

both.sides inverse==TRUEboth.sides==FALSEboth.sides==FALSEXinverse==FALSEX

X

S1.setup

gam.setup

S1.setup(G,cholesky=FALSE,no.repara=FALSE,sparse=FALSE,keepS=FALSE)

G gam.setup

cholesky

no.repara TRUE

sparse

keepS S0


```

S1[[b]]
  reparaFALSE
  start, stopstart:stop
  Sdim = stop-start+1length(S)==1rSSrepara==TRUErS
  rS
  Dstart:stopX[,start:stop]%%diag(D)X[,start:stop]b.orig = D*b.reparab.orig
  X[,start:stop]%%DX[,start:stop]b.orig = D%%b.reparab.orig
  S0

```

slanczos

```
slanczos(A,k=10,kl=-1,tol=.Machine$double.eps^.5,nt=1)
```

```

A
k      klkk1kk1
kl      klklkk1k
tol      tol
nt

```

```

klkk1klk
Akk1

```

```
valuesvectorsiter
```

```
<simon.wood@r-project.org>
```

[tprs](#)

```

require(mgcv)
## create some x's and knots...
set.seed(1);
n <- 700; A <- matrix(runif(n*n),n,n); A <- A+t(A)

## compare timings of slanczos and eigen
system.time(er <- slanczos(A,10))
system.time(um <- eigen(A,symmetric=TRUE))

## confirm values are the same...
ind <- c(1:6,(n-3):n)
range(er$values-um$values[ind]); range(abs(er$vectors)-abs(um$vectors[,ind]))

```

smooth.construct

```

xx.smooth.specsmooth.constructsmooth.constructPredict.matrix
smooth.constructsmooth.construct2smoothConbysmoothCon

```

```

smooth.construct(object,data,knots)
smooth.construct2(object,data,knots)

```

object	<pre> ste sxx.smooth.spec xbssobject tensor.smooth.spec kscCBS.dim<0 TRUEFALSE NAmsobject bys"NA" "tp" NULL NULLspgam objecttensor.smooth.specte by"NA" TRUE TRUE TRUE NULL NULLspgam </pre>
data	<pre> smooth.constructobject\$termobject\$termbyobject\$by"NA" smooth.construct2dataobject\$term </pre>
knots	<pre> object\$termNULLknotsdatsmooth.constructsmooth.construct2 </pre>

tp.smooth.specprsts.smooth.speccr.smooth.speccubic.regression.spline
cs.smooth.speccc.smooth.specps.smooth.specp.splinecp.smooth.specad.smooth.spec
adaptive.smoothre.smooth.specmrf.smooth.spectensor.smooth.spec

mgcv.smoothmgcv::plot.sos.smoothmgcv::plot.random.effect
mgcv::plot.mgcv.smooth

object

X "offset"nrow(X)byby.doneobject

S

rank

null.space.dim

C NULLsmoothConC"always.apply"byC

Cp

no.rescale gamm

df smoothConssmoothCon

te.ok 0121NULL

plot.me FALSEplot.gamTRUENULL

side.constrain FALSE

L SLS[[i]]L=NULLS[[i]]

Predict.matrixDetails

tensor.smoothmargintensor.smoothsXPNULL

"qrc""nCons"

<simon.wood@r-project.org>

sget.vargammgamPredict.matrixsmoothConPredictMat

```

## Adding a penalized truncated power basis class and methods
## as favoured by Ruppert, Wand and Carroll (2003)
## Semiparametric regression CUP. (No advantage to actually
## using this, since mgcv can happily handle non-identity
## penalties.)

smooth.construct.tr.smooth.spec<-function(object,data,knots) {
  ## a truncated power spline constructor method function
  ## object$p.order = null space dimension
  m <- object$p.order[1]
  if (is.na(m)) m <- 2 ## default
  if (m<1) stop("silly m supplied")
  if (object$bs.dim<0) object$bs.dim <- 10 ## default
  nk<-object$bs.dim-m-1 ## number of knots
  if (nk<=0) stop("k too small for m")
  x <- data[[object$term]] ## the data
  x.shift <- mean(x) # shift used to enhance stability
  k <- knots[[object$term]] ## will be NULL if none supplied
  if (is.null(k)) # space knots through data
  { n<-length(x)
    k<-quantile(x[2:(n-1)],seq(0,1,length=nk+2))[2:(nk+1)]
  }
  if (length(k)!=nk) # right number of knots?
  stop(paste("there should be ",nk," supplied knots"))
  x <- x - x.shift # basis stabilizing shift
  k <- k - x.shift # knots treated the same!
  X<-matrix(0,length(x),object$bs.dim)
  for (i in 1:(m+1)) X[,i] <- x^(i-1)
  for (i in 1:nk) X[,i+m+1]<-(x-k[i])^m*as.numeric(x>k[i])
  object$X<-X # the finished model matrix
  if (!object$fixed) # create the penalty matrix
  { object$S[[1]]<-diag(c(rep(0,m+1),rep(1,nk)))
  }
  object$rank<-nk # penalty rank
  object$null.space.dim <- m+1 # dim. of unpenalized space
  ## store "tr" specific stuff ...
  object$knots<-k;object$m<-m;object$x.shift <- x.shift

  object$df<-ncol(object$X) # maximum DoF (if unconstrained)

  class(object)<-"tr.smooth" # Give object a class
  object
}

Predict.matrix.tr.smooth<-function(object,data) {
  ## prediction method function for the `tr' smooth class
  x <- data[[object$term]]
  x <- x - object$x.shift # stabilizing shift
  m <- object$m; # spline order (3=cubic)
  k<-object$knots # knot locations
  nk<-length(k) # number of knots
  X<-matrix(0,length(x),object$bs.dim)
  for (i in 1:(m+1)) X[,i] <- x^(i-1)
  for (i in 1:nk) X[,i+m+1] <- (x-k[i])^m*as.numeric(x>k[i])
  X # return the prediction matrix
}

```

```
# an example, using the new class...
require(mgcv)
set.seed(100)
dat <- gamSim(1,n=400,scale=2)
b<-gam(y~s(x0,bs="tr",m=2)+s(x1,bs="ps",m=c(1,3))+
      s(x2,bs="tr",m=3)+s(x3,bs="tr",m=2),data=dat)
plot(b,pages=1)
b<-gamm(y~s(x0,bs="tr",m=2)+s(x1,bs="ps",m=c(1,3))+
      s(x2,bs="tr",m=3)+s(x3,bs="tr",m=2),data=dat)
plot(b$gam,pages=1)
# another example using tensor products of the new class
dat <- gamSim(2,n=400,scale=.1)$data
b <- gam(y~te(x,z,bs=c("tr","tr"),m=c(2,2)),data=dat)
vis.gam(b)
```

smooth.construct.ad.smooth.spec

```
gams(...,bs="ad",...)gammAdaptFit
knkmm
txt$bs"ps""cp""cc""cr"
m=10
scasmsmooth.construct.sc.smooth.spec
```

```
## S3 method for class 'ad.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'scad.smooth.spec'
smooth.construct(object, data, knots)
```

```
object      s(...,bs="ad",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
gamsmooth.construct2
gamm
```

```
"pspline.smooth""tensor.smooth"
```

```
<simon.wood@r-project.org>
```

```

## Comparison using an example taken from AdaptFit
## library(AdaptFit)
require(mgcv)
set.seed(0)
x <- 1:1000/1000
mu <- exp(-400*(x-.6)^2)+5*exp(-500*(x-.75)^2)/3+2*exp(-500*(x-.9)^2)
y <- mu+0.5*rnorm(1000)

##fit with default knots
## y.fit <- asp(y~f(x))

par(mfrow=c(2,2))
## plot(y.fit,main=round(cor(fitted(y.fit),mu),digits=4))
## lines(x,mu,col=2)

b <- gam(y~s(x,bs="ad",k=40,m=5)) ## adaptive
plot(b,shade=TRUE,main=round(cor(fitted(b),mu),digits=4))
lines(x,mu-mean(mu),col=2)

b <- gam(y~s(x,k=40)) ## non-adaptive
plot(b,shade=TRUE,main=round(cor(fitted(b),mu),digits=4))
lines(x,mu-mean(mu),col=2)

b <- gam(y~s(x,bs="ad",k=40,m=5,xt=list(bs="cr")))
plot(b,shade=TRUE,main=round(cor(fitted(b),mu),digits=4))
lines(x,mu-mean(mu),col=2)

## A 2D example (marked, 'Not run' purely to reduce
## checking load on CRAN).

par(mfrow=c(2,2),mar=c(1,1,1,1))
x <- seq(-.5, 1.5, length= 60)
z <- x
f3 <- function(x,z,k=15) { r<-sqrt(x^2+z^2);f<-exp(-r^2*k);f}
f <- outer(x, z, f3)
op <- par(bg = "white")

## Plot truth...
persp(x,z,f,theta=30,phi=30,col="lightblue",ticktype="detailed")

n <- 2000
x <- runif(n)*2-.5
z <- runif(n)*2-.5
f <- f3(x,z)
y <- f + rnorm(n)*.1

## Try tprs for comparison...
b0 <- gam(y~s(x,z,k=150))
vis.gam(b0,theta=30,phi=30,ticktype="detailed")

## Tensor product with non-adaptive version of adaptive penalty
b1 <- gam(y~s(x,z,bs="ad",k=15,m=1),gamma=1.4)
vis.gam(b1,theta=30,phi=30,ticktype="detailed")

## Now adaptive...
b <- gam(y~s(x,z,bs="ad",k=15,m=3),gamma=1.4)

```

```
vis.gam(b,theta=30,phi=30,ticktype="detailed")
cor(fitted(b0),f);cor(fitted(b),f)
```

```
smooth.construct.bs.smooth.spec
```

```
gam(s(x,bs="bs",m=c(3,2))m[1]m[1]=3m[1]=2m[2]m=c(3,2)mmm[1]m[2]=m[1]-1m=3m[1]
tpersp.spline
"sc"scasm
```

```
## S3 method for class 'bs.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'sc.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'Bspline.smooth'
Predict.matrix(object, data)
```

```
object      s(x,bs="bs",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
m[2]>m[1]s(x,bs="bs",m=c(3,2,1,0))
km[1]m[1]k + m[1] + 1k-m[1]+1
```

```
k-m[1]+1
```

```
deriv
```

```
scasms(x,bs="sc",xt="m+")s(x,bs="sc",xt=c("m+","c-"))xt+"m+"m-"c+"c-"scasm
```

```
"Bspline.smooth"smooth.construct
```

```
m[1]
```

```
<simon.wood@r-project.org>
```

<https://arxiv.org/abs/1605.02446>

p.spline

```
require(mgcv)
set.seed(5)
dat <- gamSim(1,n=400,dist="normal",scale=2)
bs <- "bs"
## note the double penalty on the s(x2) term...
b <- gam(y~s(x0,bs=bs,m=c(4,2))+s(x1,bs=bs)+s(x2,k=15,bs=bs,m=c(4,3,0))+
        s(x3,bs=bs,m=c(1,0)),data=dat,method="REML")
plot(b,pages=1)

## Extrapolation example, illustrating the importance of considering
## the penalty carefully if extrapolating...
f3 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 * (10 * x)^3 *
        (1 - x)^10 ## test function
n <- 100;x <- runif(n)
y <- f3(x) + rnorm(n)*2
## first a model with first order penalty over whole real line (red)
b0 <- gam(y~s(x,m=1,k=20),method="ML")
## now a model with first order penalty evaluated over (-.5,1.5) (black)
op <- options(warn=-1)
b <- gam(y~s(x,bs="bs",m=c(3,1),k=20),knots=list(x=c(-.5,0,1,1.5)),
        method="ML")
options(op)
## and the equivalent with same penalty over data range only (blue)
b1 <- gam(y~s(x,bs="bs",m=c(3,1),k=20),method="ML")
pd <- data.frame(x=seq(-.7,1.7,length=200))
fv <- predict(b,pd,se=TRUE)
ul <- fv$fit + fv$se.fit*2; ll <- fv$fit - fv$se.fit*2
plot(x,y,xlim=c(-.7,1.7),ylim=range(c(y,ll,ul)),main=
     "Order 1 penalties: red tps; blue bs on (0,1); black bs on (-.5,1.5)")
## penalty defined on (-.5,1.5) gives plausible predictions and intervals
## over this range...
lines(pd$x,fv$fit);lines(pd$x,ul,lty=2);lines(pd$x,ll,lty=2)
fv <- predict(b0,pd,se=TRUE)
ul <- fv$fit + fv$se.fit*2; ll <- fv$fit - fv$se.fit*2
## penalty defined on whole real line gives constant width intervals away
## from data, as slope there must be zero, to avoid infinite penalty:
lines(pd$x,fv$fit,col=2)
lines(pd$x,ul,lty=2,col=2);lines(pd$x,ll,lty=2,col=2)
fv <- predict(b1,pd,se=TRUE)
ul <- fv$fit + fv$se.fit*2; ll <- fv$fit - fv$se.fit*2
## penalty defined only over the data interval (0,1) gives wild and wide
## extrapolation since penalty has been `turned off' outside data range:
lines(pd$x,fv$fit,col=4)
lines(pd$x,ul,lty=2,col=4);lines(pd$x,ll,lty=2,col=4)

## construct smooth of x. Model matrix sm$X and penalty
## matrix sm$S[[1]] will have many zero entries...
x <- seq(0,1,length=100)
sm <- smoothCon(s(x,bs="bs"),data.frame(x))[[1]]

## another example checking penalty numerically...
m <- c(4,2); k <- 15; b <- runif(k)
```



```

sm <- smoothCon(s(x,bs="bs",m=m,k=k),data.frame(x),
  scale.penalty=FALSE)[[1]]
sm$deriv <- m[2]
h0 <- 1e-3; xk <- sm$knots[(m[1]+1):(k+1)]
Xp <- PredictMat(sm,data.frame(x=seq(xk[1]+h0/2,max(xk)-h0/2,h0)))
sum((Xp%*%b)^2*h0) ## numerical approximation to penalty
b%*%sm$S[[1]]%*%b ## `exact' version

## ...repeated with uneven knot spacing...
m <- c(4,2); k <- 15; b <- runif(k)
## produce the required 20 unevenly spaced knots...
knots <- data.frame(x=c(-.4,-.3,-.2,-.1,-.001,.05,.15,
  .21,.3,.32,.4,.6,.65,.75,.9,1.001,1.1,1.2,1.3,1.4))
sm <- smoothCon(s(x,bs="bs",m=m,k=k),data.frame(x),
  knots=knots,scale.penalty=FALSE)[[1]]
sm$deriv <- m[2]
h0 <- 1e-3; xk <- sm$knots[(m[1]+1):(k+1)]
Xp <- PredictMat(sm,data.frame(x=seq(xk[1]+h0/2,max(xk)-h0/2,h0)))
sum((Xp%*%b)^2*h0) ## numerical approximation to penalty
b%*%sm$S[[1]]%*%b ## `exact' version

```

smooth.construct.cr.smooth.spec

[gams](#)(x,bs="cr")s(x,bs="cs")s(x,bs="cc")

[kte](#)

k

```

## S3 method for class 'cr.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'cs.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'cc.smooth.spec'
smooth.construct(object, data, knots)

```

object	s(...,bs="cr",...)s(...,bs="cs",...)s(...,bs="cc",...)
data	byobject\$termobject\$byby
knots	dataNULL

[gamsmooth.construct2](#)

xx

```
"cr.smooth""cs.smooth""cyclic.smooth"smooth.construct
```

```
xp
```

```
F          "cr.smooth""cs.smooth"t(F)
```

```
BD          "cyclic.smooth"BD
```

```
<simon.wood@r-project.org>
```

```
## cyclic spline example...
require(mgcv)
set.seed(6)
x <- sort(runif(200)*10)
z <- runif(200)
f <- sin(x*2*pi/10)+.5
y <- rpois(exp(f),exp(f))

## finished simulating data, now fit model...
b <- gam(y ~ s(x,bs="cc",k=12) + s(z),family=poisson,
         knots=list(x=seq(0,10,length=12)))

## or more simply
b <- gam(y ~ s(x,bs="cc",k=12) + s(z),family=poisson,
         knots=list(x=c(0,10)))

## plot results...
par(mfrow=c(2,2))
plot(x,y);plot(b,select=1,shade=TRUE);lines(x,f-mean(f),col=2)
plot(b,select=2,shade=TRUE);plot(fitted(b),residuals(b))
```

```
smooth.construct.ds.smooth.spec
```

```
s(x,z,bs="ds",m=c(1,.5)gamm[1]m[2])
```

```
tprs
```

```
## S3 method for class 'ds.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'duchon.spline'
Predict.matrix(object, data)
```

```
object      s(...,bs="ds",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
k=M+k.defMk.def
gamsmooth.construct2
objecttxtsmax.knotsseed
knotstpkmax.knotsk
```

```
"duchon.spline"smooth.construct

shift
Xu
UZ
null.space.dimension
```

```
<simon.wood@r-project.org>
```

[Spherical.Spline](#)

```
require(mgcv)
eg <- gamSim(2,n=200,scale=.05)
attach(eg)
op <- par(mfrow=c(2,2),mar=c(4,4,1,1))
b0 <- gam(y~s(x,z,bs="ds",m=c(2,0),k=50),data=data) ## tps
b <- gam(y~s(x,z,bs="ds",m=c(1,.5),k=50),data=data) ## first deriv penalty
b1 <- gam(y~s(x,z,bs="ds",m=c(2,.5),k=50),data=data) ## modified 2nd deriv

persp(truth$x,truth$z,truth$f,theta=30) ## truth
vis.gam(b0,theta=30)
vis.gam(b,theta=30)
vis.gam(b1,theta=30)

detach(eg)
```

```
smooth.construct.fs.smooth.spec
```

```
gamby
```

```
factor.smooth
```

```
## S3 method for class 'fs.smooth.spec'  
smooth.construct(object, data, knots)  
## S3 method for class 'fs.interaction'  
Predict.matrix(object, data)
```

```
object      smooth.constructs(x,...,bs="fs",)gammpredict.Matrix  
            "fs.interaction"smooth.construct  
  
data        byobject$term  
  
knots
```

```
gams(x,fac,bs="fs")s(x,by=fac,id=1)gamselect=TRUE  
gammgamms(x)+s(z,fac,bs="fs")+s(v,fac,bs="fs")s(x)+s(z,fac1,bs="fs")+s(v,fac2,bs="fs")  
gamm"gammm"  
gamm4gamm4gamm"fs""gammm"  
"tp"xtss(x,fac,bs="fs",xt="cr")s(x,fac,bs="fs",xt=list(bs="cr"))ks(...,bs="fs")  
gammgamm4k  
scheme==0scheme==1
```

```
"fs.interaction""fs.interaction"smooth.constructgamm
```

```
<simon.wood@r-project.org>
```

```
factor.smoothgammsmooth.construct.sz.smooth.spec
```

```
library(mgcv)  
set.seed(0)  
## simulate data...  
f0 <- function(x) 2 * sin(pi * x)  
f1 <- function(x,a=2,b=-1) exp(a * x)+b  
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *  
      (10 * x)^3 * (1 - x)^10  
n <- 500;nf <- 25  
fac <- sample(1:nf,n,replace=TRUE)
```

```

x0 <- runif(n); x1 <- runif(n); x2 <- runif(n)
a <- rnorm(nf)*.2 + 2; b <- rnorm(nf)*.5
f <- f0(x0) + f1(x1,a[fac],b[fac]) + f2(x2)
fac <- factor(fac)
y <- f + rnorm(n)*2
## so response depends on global smooths of x0 and
## x2, and a smooth of x1 for each level of fac.

## fit model...
bm <- gamm(y~s(x0)+ s(x1,fac,bs="fs",k=5)+s(x2,k=20))
plot(bm$gam,pages=1)
summary(bm$gam)

## Also efficient using bam(..., discrete=TRUE)
bd <- bam(y~s(x0)+ s(x1,fac,bs="fs",k=5)+s(x2,k=20),discrete=TRUE)
plot(bd,pages=1)
summary(bd)

## Could also use...
## b <- gam(y~s(x0)+ s(x1,fac,bs="fs",k=5)+s(x2,k=20),method="ML")
## ... but its slower (increasingly so with increasing nf)
## b <- gam(y~s(x0)+ t2(x1,fac,bs=c("tp","re"),k=5,full=TRUE)+
##           s(x2,k=20),method="ML")
## ... is exactly equivalent.

```

smooth.construct.gp.smooth.spec

```

s(...,bs="gp")gammNAabs(m[1])m[1]m[2]m[3]

```

```

## S3 method for class 'gp.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'gp.smooth'
Predict.matrix(object, data)

```

```

object      s(...,bs="ms",...)
data        byobject$termobject$byby
knots       dataNULL

```

$\rho > 00 < \kappa \leq 2dm[1]$

$$1 - 1.5d/\rho + 0.5(d/\rho)^3d \leq \rho$$

$$\exp(-(d/\rho)^\kappa)$$

$$\exp(-d/\rho)(1 + d/\rho)$$

$$\exp(-d/\rho)(1 + d/\rho + (d/\rho)^2/3)$$

$$\exp(-d/\rho)(1 + d/\rho + 2(d/\rho)^2/5 + (d/\rho)^3/15)$$

```

r
k=M+k.defMk.def
gamsmooth.construct2
objecttxtsmax.knotsseed
knotstpkmax.knotsk

"gp.smooth"smooth.construct

shift
Xu
UZ
null.space.dimension

gp.defn

<simon.wood@r-project.org>

```

tprs

```

require(mgcv)
eg <- gamSim(2,n=200,scale=.05)
attach(eg)
op <- par(mfrow=c(2,2),mar=c(4,4,1,1))
b0 <- gam(y~s(x,z,k=50),data=data) ## tps
b <- gam(y~s(x,z,bs="gp",k=50),data=data) ## Matern spline default range
b1 <- gam(y~s(x,z,bs="gp",k=50,m=c(1,.5)),data=data) ## spherical

persp(truth$x,truth$z,truth$f,theta=30) ## truth
vis.gam(b0,theta=30)
vis.gam(b,theta=30)
vis.gam(b1,theta=30)

## compare non-stationary (b1) and stationary (b2)
b2 <- gam(y~s(x,z,bs="gp",k=50,m=c(-1,.5)),data=data) ## sph stationary
vis.gam(b1,theta=30);vis.gam(b2,theta=30)
x <- seq(-1,2,length=200); z <- rep(.5,200)
pd <- data.frame(x=x,z=z)

```

```
plot(x,predict(b1,pd),type="l");lines(x,predict(b2,pd),col=2)
abline(v=c(0,1))
plot(predict(b1),predict(b2))

detach(eg)
```

```
smooth.construct.mrf.smooth.spec
```

```
mgcv
```

```
## S3 method for class 'mrf.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'mrf.smooth'
Predict.matrix(object, data)
```

```
object      smooth.constructs(x,...,bs="mrf",xt=list(polys=foo))xxt
             Predict.Matrix"mrf.smooth"smooth.construct
data        byobject$termobject$byby
knots
```

```
xtspolysnbpenalty
names(polys)polys[[i]]NApolys[[i]]columb.polys
  polys
  polysxt
names(nb)nb[[i]]inbnames(nb)nb[[i]]
  penalty
```

```
in.outscheme==0scheme==1
drop.unused.levels=FALSEgambangamm
```

```
"mrf.smooth"data
```

```
<simon.wood@r-project.org>
```

`in.outpolys.plot`

```
library(mgcv)
## Load Columbus Ohio crime data (see ?columbus for details and credits)
data(columb)      ## data frame
data(columb.polys) ## district shapes list
xt <- list(polys=columb.polys) ## neighbourhood structure info for MRF
par(mfrow=c(2,2))
## First a full rank MRF...
b <- gam(crime ~ s(district,bs="mrf",xt=xt),data=columb,method="REML")
plot(b,scheme=1)
## Compare to reduced rank version...
b <- gam(crime ~ s(district,bs="mrf",k=20,xt=xt),data=columb,method="REML")
plot(b,scheme=1)
## An important covariate added...
b <- gam(crime ~ s(district,bs="mrf",k=20,xt=xt)+s(income),
          data=columb,method="REML")
plot(b,scheme=c(0,1))

## plot fitted values by district
par(mfrow=c(1,1))
fv <- fitted(b)
names(fv) <- as.character(columb$district)
polys.plot(columb.polys,fv)

## Examine an example neighbourhood list - this one auto-generated from
## 'polys' above.

nb <- b$smooth[[1]]$xt$nb
head(nb)
names(nb) ## these have to match the factor levels of the smooth
## look at the indices of the neighbours of the first entry,
## named '0'...
nb[['0']] ## by name
nb[[1]]   ## same by index
## ... and get the names of these neighbours from their indices...
names(nb)[nb[['0']]]
b1 <- gam(crime ~ s(district,bs="mrf",k=20,xt=list(nb=nb))+s(income),
          data=columb,method="REML")
b1 ## fit unchanged
plot(b1) ## but now there is no information with which to plot the mrf
```

`smooth.construct.ps.smooth.spec`

`gams(x,bs="ps")s(x,bs="cp",...)te`
`d.spline"tp""cr"`


```
## S3 method for class 'ps.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'cp.smooth.spec'
smooth.construct(object, data, knots)
```

```
object      s(x,bs="ps",...)s(x,bs="cp",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
s(x,bs="ps",m=c(2,3))mm=c(2,2)
km[1]+1"ps"m[1]"cp"m[1]+1m[1]
k+1"cp""ps"k + m[1] + 2k-m[1]
```

```
k-m[1]
"ps"mgcvscamderiv
```

```
"pspline.smooth""cp.smooth"smooth.construct
```

```
<simon.wood@r-project.org>
```

[cSplineDesadaptive.smoothd.spline](#)

```
## see ?gam
## cyclic example ...
require(mgcv)
set.seed(6)
x <- sort(runif(200)*10)
z <- runif(200)
f <- sin(x*2*pi/10)+.5
y <- rpois(exp(f),exp(f))

## finished simulating data, now fit model...
b <- gam(y ~ s(x,bs="cp") + s(z,bs="ps"),family=poisson)

## example with supplied knot ranges for x and z (can do just one)
b <- gam(y ~ s(x,bs="cp") + s(z,bs="ps"),family=poisson,
        knots=list(x=c(0,10),z=c(0,1)))
```

```

## example with supplied knots...
bk <- gam(y ~ s(x,bs="cp",k=12) + s(z,bs="ps",k=13),family=poisson,
          knots=list(x=seq(0,10,length=13),z=(-3):13/10))

## plot results...
par(mfrow=c(2,2))
plot(b,select=1,shade=TRUE);lines(x,f-mean(f),col=2)
plot(b,select=2,shade=TRUE);lines(z,0*z,col=2)
plot(bk,select=1,shade=TRUE);lines(x,f-mean(f),col=2)
plot(bk,select=2,shade=TRUE);lines(z,0*z,col=2)

## Example using monotonic constraints via the SCOP-spline
## construction, and of computing derivatives...
x <- seq(0,1,length=100); dat <- data.frame(x)
sspec <- s(x,bs="ps")
sspec$mono <- 1
sm <- smoothCon(sspec,dat)[[1]]
sm$deriv <- 1
Xd <- PredictMat(sm,dat)
## generate random coefficients in the unconstrained
## parameterization...
b <- runif(10)*3-2.5
## exponentiate those parameters indicated by sm$g.index
## to obtain coefficients meeting the constraints...
b[sm$g.index] <- exp(b[sm$g.index])
## plot monotonic spline and its derivative
par(mfrow=c(2,2))
plot(x,sm$X%*%b,type="l",ylab="f(x)")
plot(x,Xd%*%b,type="l",ylab="f'(x)")
## repeat for decrease...
sspec$mono <- -1
sm1 <- smoothCon(sspec,dat)[[1]]
sm1$deriv <- 1
Xd1 <- PredictMat(sm1,dat)
plot(x,sm1$X%*%b,type="l",ylab="f(x)")
plot(x,Xd1%*%b,type="l",ylab="f'(x)")

## Now with sum to zero constraints as well...
sspec$mono <- 1
sm <- smoothCon(sspec,dat,absorb.cons=TRUE)[[1]]
sm$deriv <- 1
Xd <- PredictMat(sm,dat)
b <- b[-1] ## dropping first param
plot(x,sm$X%*%b,type="l",ylab="f(x)")
plot(x,Xd%*%b,type="l",ylab="f'(x)")

sspec$mono <- -1
sm1 <- smoothCon(sspec,dat,absorb.cons=TRUE)[[1]]
sm1$deriv <- 1
Xd1 <- PredictMat(sm1,dat)
plot(x,sm1$X%*%b,type="l",ylab="f(x)")
plot(x,Xd1%*%b,type="l",ylab="f'(x)")

```

smooth.construct.re.smooth.spec

`gams(x,bs="re")`

```
## S3 method for class 're.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'random.effect'
Predict.matrix(object, data)
```

```
object      smooth.constructs(x,...,bs="re",)predict.Matrix"random.effect"
            smooth.construct
```

```
data        byobject$termobject$byby
```

```
knots
```

```
s(x,z,bs="re")~x:z-1
```

```
s(x,bs="re")model.matrix(~x-1)s(x,v,w,bs="re")model.matrix(~x:v:w-1)
```

`linear.functional.terms`

$$\sum_j \lambda_j S_j S_j \lambda_j S_j \text{xtsrankxt} S_j$$

`id`

`gamm4gammgam`

`drop.unused.levels=FALSEgambamgamm`

`predict.gamrandom.effects``Predict.matrixNApredict.gamNA`

`"random.effect"`

`<simon.wood@r-project.org>`

`gam.vcompgamm`

```

## see ?gam.vcomp

require(mgcv)
## simulate simple random effect example
set.seed(4)
nb <- 50; n <- 400
b <- rnorm(nb)*2 ## random effect
r <- sample(1:nb,n,replace=TRUE) ## r.e. levels
y <- 2 + b[r] + rnorm(n)
r <- factor(r)
## fit model....
b <- gam(y ~ s(r,bs="re"),method="REML")
gam.vcomp(b)

## example with supplied precision matrices...
b <- c(rnorm(nb/2)*2,rnorm(nb/2)*.5) ## random effect now with 2 variances
r <- sample(1:nb,n,replace=TRUE) ## r.e. levels
y <- 2 + b[r] + rnorm(n)
r <- factor(r)
## known precision matrix components...
S <- list(diag(rep(c(1,0),each=nb/2)),diag(rep(c(0,1),each=nb/2)))
b <- gam(y ~ s(r,bs="re",xt=list(S=S,rank=c(nb/2,nb/2))),method="REML")
gam.vcomp(b)
summary(b)

```

smooth.construct.so.smooth.spec

sosfswgammgamm4smooth.construct2smoothCon

```

## S3 method for class 'so.smooth.spec'
smooth.construct(object,data,knots)
## S3 method for class 'sf.smooth.spec'
smooth.construct(object,data,knots)
## S3 method for class 'sw.smooth.spec'
smooth.construct(object,data,knots)

```

```

object      s(...,bs="so",xt=list(bnd=bnd,...))gamxtsxtks
data
knots       ks

```

kxtsknotsknotsgam

bndxtf

bndSpecxt

"cc""cp""cc"m

locatorbnd <-as.data.frame(locator(type="l"))

nmaxxtsgam

$f(x,y)$

$$f_{xx} + f_{yy} = g$$

$f = ssg$

$$g_{xx} + g_{yy} = 0$$

$g = 0g(x_k,y_k) = c_kc_k$

$f c_k$

$$\|z - f\|^2 + \lambda_s J_s(s) + \lambda_f J_f(f)$$

$J_s J_f (f_x x + f_y y)^2 \lambda z f$

object

sd

x "offset"

s

irng

smooth.construct

<simon.wood@r-project.org>

Predict.matrix.soap.film

```

require(mgcv)

#####
## simple test function...
#####

fsb <- list(fs.boundary())
nmax <- 100
## create some internal knots...
knots <- data.frame(v=rep(seq(-.5,3,by=.5),4),
                    w=rep(c(-.6,-.3,.3,.6),rep(8,4)))
## Simulate some fitting data, inside boundary...
set.seed(0)
n<-600
v <- runif(n)*5-1;w<-runif(n)*2-1
y <- fs.test(v,w,b=1)
names(fsb[[1]]) <- c("v","w")
ind <- inSide(fsb,x=v,y=w) ## remove outsiders
y <- y + rnorm(n)*.3 ## add noise
y <- y[ind];v <- v[ind]; w <- w[ind]
n <- length(y)

par(mfrow=c(3,2))
## plot boundary with knot and data locations
plot(fsb[[1]]$v,fsb[[1]]$w,type="l");points(knots,pch=20,col=2)
points(v,w,pch=".");

## Now fit the soap film smoother. 'k' is dimension of boundary smooth.
## boundary supplied in 'xt', and knots in 'knots'...

nmax <- 100 ## reduced from default for speed.
b <- gam(y~s(v,w,k=30,bs="so",xt=list(bnd=fsb,nmax=nmax)),knots=knots)

plot(b) ## default plot
plot(b,scheme=1)
plot(b,scheme=2)
plot(b,scheme=3)

vis.gam(b,plot.type="contour")

#####
# Fit same model in two parts...
#####

par(mfrow=c(2,2))
vis.gam(b,plot.type="contour")

b1 <- gam(y~s(v,w,k=30,bs="sf",xt=list(bnd=fsb,nmax=nmax))+
          s(v,w,k=30,bs="sw",xt=list(bnd=fsb,nmax=nmax)),knots=knots)
vis.gam(b,plot.type="contour")
plot(b1)

#####
## Now an example with known boundary condition...
#####

```

```

## Evaluate known boundary condition at boundary nodes...
fsb[[1]]$f <- fs.test(fsb[[1]]$v,fsb[[1]]$w,b=1,exclude=FALSE)

## Now fit the smooth...
bk <- gam(y~s(v,w,bs="so",xt=list(bnd=fsb,nmax=nmax)),knots=knots)
plot(bk) ## default plot

#####
## tensor product example...
#####

set.seed(9)
n <- 10000
v <- runif(n)*5-1;w<-runif(n)*2-1
t <- runif(n)
y <- fs.test(v,w,b=1)
y <- y + 4.2
y <- y^(.5+t)
fsb <- list(fs.boundary())
names(fsb[[1]]) <- c("v","w")
ind <- inside(fsb,x=v,y=w) ## remove outsiders
y <- y[ind];v <- v[ind]; w <- w[ind]; t <- t[ind]
n <- length(y)
y <- y + rnorm(n)*.05 ## add noise
knots <- data.frame(v=rep(seq(-.5,3,by=.5),4),
                    w=rep(c(-.6,-.3,.3,.6),rep(8,4)))

## notice NULL element in 'xt' list - to indicate no xt object for "cr" basis...
bk <- gam(y~ te(v,w,t,bs=c("sf","cr"),k=c(25,4),d=c(2,1),
xt=list(list(bnd=fsb,nmax=nmax),NULL))+
te(v,w,t,bs=c("sw","cr"),k=c(25,4),d=c(2,1),
xt=list(list(bnd=fsb,nmax=nmax),NULL)),knots=knots)

par(mfrow=c(3,2))
m<-100;n<-50
xm <- seq(-1,3.5,length=m);yn<-seq(-1,1,length=n)
xx <- rep(xm,n);yy<-rep(yn,rep(m,n))
tru <- matrix(fs.test(xx,yy),m,n)+4.2 ## truth

image(xm,yn,tru^.5,col=heat.colors(100),xlab="v",ylab="w",
      main="truth")
lines(fsb[[1]]$v,fsb[[1]]$w,lwd=3)
contour(xm,yn,tru^.5,add=TRUE)

vis.gam(bk,view=c("v","w"),cond=list(t=0),plot.type="contour")

image(xm,yn,tru,col=heat.colors(100),xlab="v",ylab="w",
      main="truth")
lines(fsb[[1]]$v,fsb[[1]]$w,lwd=3)
contour(xm,yn,tru,add=TRUE)

vis.gam(bk,view=c("v","w"),cond=list(t=.5),plot.type="contour")

image(xm,yn,tru^1.5,col=heat.colors(100),xlab="v",ylab="w",
      main="truth")
lines(fsb[[1]]$v,fsb[[1]]$w,lwd=3)
contour(xm,yn,tru^1.5,add=TRUE)

```

```

vis.gam(bk,view=c("v","w"),cond=list(t=1),plot.type="contour")

#####
# nested boundary example...
#####

bnd <- list(list(x=0,y=0),list(x=0,y=0))
seq(0,2*pi,length=100) -> theta
bnd[[1]]$x <- sin(theta);bnd[[1]]$y <- cos(theta)
bnd[[2]]$x <- .3 + .3*sin(theta);
bnd[[2]]$y <- .3 + .3*cos(theta)
plot(bnd[[1]]$x,bnd[[1]]$y,type="l")
lines(bnd[[2]]$x,bnd[[2]]$y)

## setup knots
k <- 8
xm <- seq(-1,1,length=k);ym <- seq(-1,1,length=k)
x=rep(xm,k);y=rep(ym,rep(k,k))
ind <- inSide(bnd,x,y)
knots <- data.frame(x=x[ind],y=y[ind])
points(knots$x,knots$y)

## a test function

f1 <- function(x,y) {
  exp(-(x-.3)^2-(y-.3)^2)
}

## plot the test function within the domain
par(mfrow=c(2,3))
m<-100;n<-100
xm <- seq(-1,1,length=m);yn<-seq(-1,1,length=n)
x <- rep(xm,n);y<-rep(yn,rep(m,n))
ff <- f1(x,y)
ind <- inSide(bnd,x,y)
ff[!ind] <- NA
image(xm,yn,matrix(ff,m,n),xlab="x",ylab="y")
contour(xm,yn,matrix(ff,m,n),add=TRUE)
lines(bnd[[1]]$x,bnd[[1]]$y,lwd=2);lines(bnd[[2]]$x,bnd[[2]]$y,lwd=2)

## Simulate data by noisy sampling from test function...

set.seed(1)
x <- runif(300)*2-1;y <- runif(300)*2-1
ind <- inSide(bnd,x,y)
x <- x[ind];y <- y[ind]
n <- length(x)
z <- f1(x,y) + rnorm(n)*.1

## Fit a soap film smooth to the noisy data
nmax <- 60
b <- gam(z~s(x,y,k=c(30,15),bs="so",xt=list(bnd=bnd,nmax=nmax)),
  knots=knots,method="REML")
plot(b) ## default plot
vis.gam(b,plot.type="contour") ## prettier version

```



```
## trying out separated fits...
ba <- gam(z~s(x,y,k=c(30,15),bs="sf",xt=list(bnd=bnd,nmax=nmax))+
  s(x,y,k=c(30,15),bs="sw",xt=list(bnd=bnd,nmax=nmax)),
  knots=knots,method="REML")
plot(ba)
vis.gam(ba,plot.type="contour")
```

```
smooth.construct.sos.smooth.spec
```

```
gams(la,lo,bs="sos",m=2,k=100)mm>0(m+2)/2m=2m=0m = -1Duchon.spline
k
```

```
## S3 method for class 'sos.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'sos.smooth'
Predict.matrix(object, data)
```

```
object      s(...,bs="sos",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
m>0m=0m = -1m = -2
```

```
scheme==0thetaphi
scheme==1scheme==0scheme>1scheme
```

```
"sos.smooth"smooth.construct
```

```
Xu
```

```
UZ
```

```
<simon.wood@r-project.org>
```

Duchon.spline

```
require(mgcv)
set.seed(0)
n <- 400

f <- function(la,lo) { ## a test function...
  sin(lo)*cos(la-.3)
}

## generate with uniform density on sphere...
lo <- runif(n)*2*pi-pi ## longitude
la <- runif(3*n)*pi-pi/2
ind <- runif(3*n)<=cos(la)
la <- la[ind];
la <- la[1:n]

ff <- f(la,lo)
y <- ff + rnorm(n)*.2 ## test data

## generate data for plotting truth...
lam <- seq(-pi/2,pi/2,length=30)
lom <- seq(-pi,pi,length=60)
gr <- expand.grid(la=lam,lo=lom)
fz <- f(gr$la,gr$lo)
zm <- matrix(fz,30,60)

require(mgcv)
dat <- data.frame(la = la *180/pi, lo = lo *180/pi, y=y)

## fit spline on sphere model...
bp <- gam(y~s(la,lo,bs="sos",k=60),data=dat)

## pure knot based alternative...
ind <- sample(1:n,100)
bk <- gam(y~s(la,lo,bs="sos",k=60),
  knots=list(la=dat$la[ind],lo=dat$lo[ind]),data=dat)

b <- bp

cor(fitted(b),ff)

## plot results and truth...

pd <- data.frame(la=gr$la*180/pi,lo=gr$lo*180/pi)
fv <- matrix(predict(b,pd),30,60)

par(mfrow=c(2,2),mar=c(4,4,1,1))
contour(lom,lam,t(zm))
contour(lom,lam,t(fv))
plot(bp,rug=FALSE)
plot(bp,scheme=1,theta=-30,phi=20,pch=19,cex=.5)
```

```
smooth.construct.sz.smooth.spec
```

```
## S3 method for class 'sz.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'sz.interaction'
Predict.matrix(object, data)
```

```
object      smooth.constructs(x,...,bs="sz",)predict.Matrix"sz.interaction"
              smooth.construct
data         byobject$term
knots
```

```
s(fac,x,bs="sz")xfac
```

$$g(\mu_i) = f(x_i) + f_{k(i)}(x_i)$$

$$k(i)f_k\beta_{ki}f_k\sum_k\beta_{ki} = 0$$

```
tensor.prod.model.matrix
```

```
ids(fac,x,bs="sz",id=1)
```

```
xtsbss(fac,x,xt=list(bs="cr"))"cr""tp"
```

```
scheme==0scheme==1
```

```
"sz.interaction"
```

```
<simon.wood@r-project.org>
```

```
gam.modelsgamm
```

```
library(mgcv)
```

```
set.seed(0)
```

```
dat <- gamSim(4)
```

```
b <- gam(y ~ s(x2)+s(fac,x2,bs="sz")+s(x0),data=dat,method="REML")
```

```
plot(b,pages=1)
```

```
summary(b)
```

```
## Example involving 2 factors

f1 <- function(x2) 2 * sin(pi * x2)
f2 <- function(x2) exp(2 * x2) - 3.75887
f3 <- function(x2) 0.2 * x2^11 * (10 * (1 - x2))^6 + 10 * (10 * x2)^3 *
  (1 - x2)^10

n <- 600
x <- runif(n)
f1 <- factor(sample(c("a", "b", "c"), n, replace=TRUE))
f2 <- factor(sample(c("foo", "bar"), n, replace=TRUE))

mu <- f3(x)
for (i in 1:3) mu <- mu + exp(2*(2-i)*x)*(f1==levels(f1)[i])
for (i in 1:2) mu <- mu + 10*i*x*(1-x)*(f2==levels(f2)[i])
y <- mu + rnorm(n)
dat <- data.frame(y=y, x=x, f1=f1, f2=f2)
b <- gam(y ~ s(x)+s(f1, x, bs="sz")+s(f2, x, bs="sz")+s(f1, f2, x, bs="sz", id=1), data=dat, method="REML")
plot(b, pages=1, scale=0)
```

smooth.construct.t2.smooth.spec

smooth.construct

```
## S3 method for class 't2.smooth.spec'
smooth.construct(object, data, knots)
```

```
object      t2.smooth.spect2(x,z)gam
data        byobject$termobject$byby
knots       dataNULL
```

[t2tesmooth.constructsmooth.terms](#)

"t2.smooth"

<simon.wood@r-project.org>

[t2](#)

see ?t2

`smooth.construct.tensor.smooth.spec`

`smooth.construct`

S3 method for class 'tensor.smooth.spec'
`smooth.construct(object, data, knots)`

<code>object</code>	<code>tensor.smooth.specte(x,z)</code> gam
<code>data</code>	<code>byobject\$termobject\$byby</code>
<code>knots</code>	<code>dataNULL</code>

[tesmooth.constructsmooth.terms](#)

"tensor.smooth"[smooth.construct](#)

<simon.wood@r-project.org>

[cSplineDes](#)

see ?gam

smooth.construct.tp.smooth.spec

```
gams(x,z,bs="tp",m=3)s(x,z)m
m
```

```
max.knotsmax.knotsmax.knotsmax.knotsxts
"ts"
```

```
## S3 method for class 'tp.smooth.spec'
smooth.construct(object, data, knots)
## S3 method for class 'ts.smooth.spec'
smooth.construct(object, data, knots)
```

```
object      s(...,bs="tp",...)s(...,bs="ts",...)
data        byobject$termobject$byby
knots       dataNULL
```

```
k=M+k.defMk.def
m2m > d+1d2m>d
gamsmooth.construct2
objecttxttsmax.knotsseed
knotstpkmax.knotsk
```

```
"tprs.smooth""ts.smooth"smooth.construct
shift
Xu
UZ
null.space.dimension
```

<simon.wood@r-project.org>

```

require(mgcv); n <- 100; set.seed(2)
x <- runif(n); y <- x + x^2*.2 + rnorm(n) *.1

## is smooth significantly different from straight line?
summary(gam(y~s(x,m=c(2,0))+x,method="REML")) ## not quite

## is smooth significatly different from zero?
summary(gam(y~s(x),method="REML")) ## yes!

## Fool bam(...,discrete=TRUE) into (strange) nested
## model fit...
set.seed(2) ## simulate some data...
dat <- gamSim(1,n=400,dist="normal",scale=2)
dat$x1a <- dat$x1 ## copy x1 so bam allows 2 copies of x1
## Following removes identifiability problem, by removing
## linear terms from second smooth, and then re-inserting
## the one that was not a duplicate (x2)...
b <- bam(y~s(x0,x1)+s(x1a,x2,m=c(2,0))+x2,data=dat,discrete=TRUE)

## example of knot based tprs...
k <- 10; m <- 2
y <- y[order(x)]; x <- x[order(x)]
b <- gam(y~s(x,k=k,m=m),method="REML",
        knots=list(x=seq(0,1,length=k)))
X <- model.matrix(b)
par(mfrow=c(1,2))
plot(x,X[,1],ylim=range(X),type="l")
for (i in 2:ncol(X)) lines(x,X[,i],col=i)

## compare with eigen based (default)
b1 <- gam(y~s(x,k=k,m=m),method="REML")
X1 <- model.matrix(b1)
plot(x,X1[,1],ylim=range(X1),type="l")
for (i in 2:ncol(X1)) lines(x,X1[,i],col=i)
## see ?gam

```

smooth.info

smooth.info(object)

object

[bambambam](#)smooth.infosmooth.infointerpret.gam0

<simon.wood@r-project.org>

[bamsmooth.constructPredictMat](#)

```
# See smooth.construct examples
spec <- s(a,bs="re")
class(spec)
spec$tensor.possible
spec <- smooth.info(spec)
spec$tensor.possible
```

smooth.terms

[gamstetit2user.defined.smoothstetit2mgcv](#)

[gambylinear.functional.terms](#)

```
bs="tp"stprs
  bs="ts""tp"
bs="ds"Duchon.spline
bs="cr"cubic.regression.spline
  bs="cs""cr"
  bs="cc"
bs="sos"Spherical.Spline
bs="bs"
bs="ps"
  bs="cp"
bs="re"smooth.construct.re.smooth.spec
bs="mrf"mrf
bs="gp"gp.smooth
bs="so"bs="sw"bs="sf"soap
```

[tetit2](#)

```
te
ti  $f_1(x) + f_2(z) + f_3(x, z)y \sim s(x) + s(z) + ti(x, z)y \sim ti(x) + ti(z) + ti(x, z)$ te
t2gamma4
```

[tete](#)

```
bs="ad"adaptive.smoothgamma
bs="sz"bygam.modelssmooth.construct.sz.smooth.spec
bs="fs"smooth.construct.fs.smooth.spec"fs"gamma
```


<https://arxiv.org/abs/1605.02446>

```
stet2tpsrDuchon.splinecubic.regression.splinep.splined.spline  
mrfsoapSpherical.Splineadaptive.smoothuser.defined.smooth  
smooth.construct.re.smooth.specsmooth.construct.gp.smooth.spec  
factor.smooth.interaction
```

```
## see examples for gam and gamm
```

smooth2random

mgcvlme

smooth2random(object,vnames,type=1)

object	mgcv
vnames	
type	1lme1mer

mgcvlme~~gamm~~type=2

rand	type=2
trans.D	b.original = trans.U %*% (trans.D*b.fit)
trans.U	
Xf	
fixed	TRUE/FALSE
rind	
pen.ind	

<simon.wood@r-project.org>

gamm

```
## Simple type 1 'lme' style...
library(mgcv)
x <- runif(30)
sm <- smoothCon(s(x),data.frame(x=x))[[1]]
smooth2random(sm,"")

## Now type 2 'lme4' style...
z <- runif(30)
dat <- data.frame(x=x,z=z)
sm <- smoothCon(t2(x,z),dat)[[1]]
re <- smooth2random(sm,"",2)
str(re)

## For prediction after fitting we might transform parameters back to
## original parameterization using 'rind', 'trans.D' and 'trans.U',
## and call PredictMat(sm,newdata) to get the prediction matrix to
## multiply these transformed parameters by.
## Alternatively we could obtain fixed and random effect Prediction
## matrices corresponding to the results from smooth2random, which
## can be used with the fit parameters without transforming them.
## The following shows how...

s2rPred <- function(sm,re,data) {
  ## Function to aid prediction from smooths represented as type==2
  ## random effects. re must be the result of smooth2random(sm,...,type=2).
  X <- PredictMat(sm,data) ## get prediction matrix for new data
  ## transform to r.e. parameterization
  if (!is.null(re$trans.U)) X <- X%*%re$trans.U
  X <- t(t(X)*re$trans.D)
  ## re-order columns according to random effect re-ordering...
  X[,re$rind] <- X[,re$pen.ind!=0]
  ## re-order penalization index in same way
  pen.ind <- re$pen.ind; pen.ind[re$rind] <- pen.ind[pen.ind>0]
  ## start return object...
  r <- list(rand=list(),Xf=X[,which(re$pen.ind==0),drop=FALSE])
  for (i in 1:length(re$rand)) { ## loop over random effect matrices
    r$rand[[i]] <- X[,which(pen.ind==i),drop=FALSE]
    attr(r$rand[[i]],"s.label") <- attr(re$rand[[i]],"s.label")
  }
  names(r$rand) <- names(re$rand)
  r
} ## s2rPred

## use function to obtain prediction random and fixed effect matrices
## for first 10 elements of 'dat'. Then confirm that these match the
```

```
## first 10 rows of the original model matrices, as they should...

r <- s2rPred(sm,re,dat[1:10,])
range(r$Xf-re$Xf[1:10,])
range(r$rand[[1]]-re$rand[[1]][1:10,])
```

smoothCon

```
smoothCon(object,data,knots=NULL,absorb.cons=FALSE,
          scale.penalty=TRUE,n=nrow(data),dataX=NULL,
          null.space.penalty=FALSE,sparse.cons=0,
          diagonal.penalty=FALSE,apply.by=TRUE,modCon=0)
PredictMat(object,data,n=nrow(data))
```

```
object
data          n
knots
absorb.cons    TRUE
scale.penalty  gamm
n              data
dataX          dataXn
null.space.penalty

apply.by       FALSEX0bybybam
sparse.cons    0-1bam12
diagonal.penalty
              TRUEdiagRPPredictMat

modCon
```

```
smooth.constructPredict.matrixby
byby.donesmoothConPredict.matrixby"by.done"
CC
smoothConbysmoothConbyPredictMat
gammgam.control
```

```
smoothConsmoothsmooth.construct"qrc""nCons""nCons""qrc"qr
predictMatobject
```

<simon.wood@r-project.org>

[gam.controlsmooth.constructPredict.matrix](#)

```
## example of using smoothCon and PredictMat to set up a basis
## to use for regression and make predictions using the result
library(MASS) ## load for mcycle data.
## set up a smoother...
sm <- smoothCon(s(times,k=10),data=mcycle,knots=NULL)[[1]]
## use it to fit a regression spline model...
beta <- coef(lm(mcycle$accel~sm$X-1))
with(mcycle,plot(times,accel)) ## plot data
times <- seq(0,60,length=200) ## creat prediction times
## Get matrix mapping beta to spline prediction at 'times'
Xp <- PredictMat(sm,data.frame(times=times))
lines(times,Xp%*%beta) ## add smooth to plot

## Same again but using a penalized regression spline of
## rank 30....
sm <- smoothCon(s(times,k=30),data=mcycle,knots=NULL)[[1]]
E <- t(mroot(sm$S[[1]])) ## square root penalty
X <- rbind(sm$X,0.1*E) ## augmented model matrix
y <- c(mcycle$accel,rep(0,nrow(E))) ## augmented data
beta <- coef(lm(y~X-1)) ## fit penalized regression spline
Xp <- PredictMat(sm,data.frame(times=times)) ## prediction matrix
with(mcycle,plot(times,accel)) ## plot data
lines(times,Xp%*%beta) ## overlay smooth
```

sp.vcov

gam

sp.vcov(x,edge.correct=TRUE,reg=1e-3)

```
x          gamgam()
edge.correct edge.correct=TRUEgam.control
reg
```

NULL"ML""REML"edge.correct=TRUElsp

<simon.wood@r-project.org>

[gamgam.vcomp](#)

```
require(mgcv)
n <- 100
x <- runif(n); z <- runif(n)
y <- sin(x*2*pi) + rnorm(n)*.2
mod <- gam(y~s(x,bs="cc",k=10)+s(z),knots=list(x=seq(0,1,length=10)),
           method="REML")
sp.vcov(mod)
```

spasm.construct

```
spasm.construct(object,data)
spasm.sp(object,sp,w=rep(1,object$nobs),get.trH=TRUE,block=0,centre=FALSE)
spasm.smooth(object,X,residual=FALSE,block=0)
```

```
object
data
sp
w
get.trH
block
centre
X
residual
```

<simon.wood@r-project.org>

step.gam

step.gammgcvmgcv

```
"cs""ts"s  
selectgams(x,fx=TRUE)
```

<simon.wood@r-project.org>

gam.selection

```
## an example of GCV based model selection as  
## an alternative to stepwise selection, using  
## shrinkage smoothers...  
library(mgcv)  
set.seed(0);n <- 400  
dat <- gamSim(1,n=n,scale=2)  
dat$x4 <- runif(n, 0, 1)  
dat$x5 <- runif(n, 0, 1)  
attach(dat)  
## Note the increased gamma parameter below to favour  
## slightly smoother models...  
b<-gam(y~s(x0,bs="ts")+s(x1,bs="ts")+s(x2,bs="ts")+  
      s(x3,bs="ts")+s(x4,bs="ts")+s(x5,bs="ts"),gamma=1.4)  
summary(b)  
plot(b,pages=1)  
  
## Same again using REML/ML  
b<-gam(y~s(x0,bs="ts")+s(x1,bs="ts")+s(x2,bs="ts")+  
      s(x3,bs="ts")+s(x4,bs="ts")+s(x5,bs="ts"),method="REML")  
summary(b)  
plot(b,pages=1)  
  
## And once more, but using the null space penalization  
b<-gam(y~s(x0,bs="cr")+s(x1,bs="cr")+s(x2,bs="cr")+  
      s(x3,bs="cr")+s(x4,bs="cr")+s(x5,bs="cr"),  
      method="REML",select=TRUE)  
summary(b)  
plot(b,pages=1)  
  
detach(dat);rm(dat)
```

summary.gam

gamgam()[sink](#)

```
## S3 method for class 'gam'
summary(object, dispersion=NULL, freq=FALSE, re.test=TRUE, ...)

## S3 method for class 'summary.gam'
print(x,digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"),...)
```

```
object      gamgam()
x            summary.gamsummary.gam()
dispersion  NULL
freq        TRUE
re.test
digits
signif.stars
...
```

$\mathbf{A} \mathbf{P}_i \mathbf{X}^{tr} (\mathbf{X} \mathbf{P}_i)^2 \mathbf{A} - \mathbf{A} \mathbf{A}$
print.summary.gam

freq=TRUEparaPenfreq=TRUE

```
summary.gamgam

p.coeff
p.t      p.coeff
p.pv
m
chi.sq
s.pv
se
r.sq     r.sq
```

dev.expl dev.explr.sq
edf
residual.df
n
np
rank
method
sp.criterion
scale
family
formula
dispersion
pTerms.df
pTerms.chi.sq
pTerms.pv
cov.unscaled freq=TRUE
cov.scaled freq=TRUE
p.table
s.table
p.Terms

<simon.wood@r-project.org>

[gampredict.gamgam.checkanova.gamgam.vcompsp.vcov](#)


```

library(mgcv)
set.seed(0)

dat <- gamSim(1,n=200,scale=2) ## simulate data

b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b,pages=1)
summary(b)

## now check the p-values by using a pure regression spline.....
b.d <- round(summary(b)$edf)+1 ## get edf per smooth
b.d <- pmax(b.d,3) # can't have basis dimension less than 3!
bc<-gam(y~s(x0,k=b.d[1],fx=TRUE)+s(x1,k=b.d[2],fx=TRUE)+
        s(x2,k=b.d[3],fx=TRUE)+s(x3,k=b.d[4],fx=TRUE),data=dat)
plot(bc,pages=1)
summary(bc)

## Example where some p-values are less reliable...
dat <- gamSim(6,n=200,scale=2)
b <- gam(y~s(x0,m=1)+s(x1)+s(x2)+s(x3)+s(fac,bs="re"),data=dat)
## Here s(x0,m=1) can be penalized to zero, so p-value approximation
## cruder than usual...
summary(b)

## p-value check - increase k to make this useful!
k<-20;n <- 200;p <- rep(NA,k)
for (i in 1:k)
{ b<-gam(y~te(x,z),data=data.frame(y=rnorm(n),x=runif(n),z=runif(n)),
    method="ML")
  p[i]<-summary(b)$s.p[1]
}
plot(((1:k)-0.5)/k,sort(p))
abline(0,1,col=2)
ks.test(p,"punif") ## how close to uniform are the p-values?

## A Gamma example, by modify `gamSim' output...

dat <- gamSim(1,n=400,dist="normal",scale=1)
dat$f <- dat$f/4 ## true linear predictor
Ey <- exp(dat$f);scale <- .5 ## mean and GLM scale parameter
## Note that `shape' and `scale' in `rgamma' are almost
## opposite terminology to that used with GLM/GAM...
dat$y <- rgamma(Ey*0,shape=1/scale,scale=Ey*scale)
bg <- gam(y~ s(x0)+ s(x1)+s(x2)+s(x3),family=Gamma(link=log),
        data=dat,method="REML")
summary(bg)

```

t2

```
t2(..., k=NA,bs="cr",m=NA,d=NA,by=NA,xt=NULL,
      id=NULL,sp=NULL,full=FALSE,ord=NULL,pc=NULL)
```

```
...      t2(log(x),z)t2(I(x/sd(x)),z)predict.gam
k        5^dchoose.k
bs       "cr""cs""cc""tp""ts"smooth.termssmooth.construct
m        mmpp.splineDuchon.splineNA"cr"
d        d=c(2,1)
by       gam.modelsgam.modelslinear.functional.terms
xt
id       id
sp       spgamgamm
full     TRUEFALSE
ord      NULL
pc       NULLidentifiability
```

pen.edf

full=FALSEfull=TRUE

t2.smooth.specsmooth.construct.tensor.smooth.spec

```
margin    smooth.specs
term
by        by"NA"
fx        TRUEFALSE
label
dim
mp        TRUE
np        TRUE
id        idte
sp        spte
```

<simon.wood@r-project.org>

tesgamgamm

```
# following shows how tensor product deals nicely with
# badly scaled covariates (range of x 5% of range of z )
require(mgcv)
test1<-function(x,z,sx=0.3,sz=0.4)
{ x<-x*20
  (pi*sx*sz)*(1.2*exp(-(x-0.2)^2/sx^2-(z-0.3)^2/sz^2)+
    0.8*exp(-(x-0.7)^2/sx^2-(z-0.8)^2/sz^2))
}
n<-500
old.par<-par(mfrow=c(2,2))
x<-runif(n)/20;z<-runif(n);
xs<-seq(0,1,length=30)/20;zs<-seq(0,1,length=30)
pr<-data.frame(x=rep(xs,30),z=rep(zs,rep(30,30)))
truth<-matrix(test1(pr$x,pr$z),30,30)
f <- test1(x,z)
y <- f + rnorm(n)*0.2
b1<-gam(y~s(x,z))
persp(xs,zs,truth);title("truth")
vis.gam(b1);title("t.p.r.s")
b2<-gam(y~t2(x,z))
vis.gam(b2);title("tensor product")
b3<-gam(y~t2(x,z,bs=c("tp","tp")))
vis.gam(b3);title("tensor product")
par(old.par)

test2<-function(u,v,w,sv=0.3,sw=0.4)
{ ((pi*sv*sw)*(1.2*exp(-(v-0.2)^2/sv^2-(w-0.3)^2/sw^2)+
  0.8*exp(-(v-0.7)^2/sv^2-(w-0.8)^2/sw^2)))*(u-0.5)^2*20
}
n <- 500
v <- runif(n);w<-runif(n);u<-runif(n)
f <- test2(u,v,w)
y <- f + rnorm(n)*0.2

## tensor product of 2D Duchon spline and 1D cr spline
m <- list(c(1,.5),0)
b <- gam(y~t2(v,w,u,k=c(30,5),d=c(2,1),bs=c("ds","cr"),m=m))
```

```
## look at the edf per penalty. "rr" denotes interaction term
## (range space range space). "rn" is interaction of null space
## for u with range space for v,w...
pen.edf(b)

## plot results...
op <- par(mfrow=c(2,2))
vis.gam(b,cond=list(u=0),color="heat",zlim=c(-0.2,3.5))
vis.gam(b,cond=list(u=.33),color="heat",zlim=c(-0.2,3.5))
vis.gam(b,cond=list(u=.67),color="heat",zlim=c(-0.2,3.5))
vis.gam(b,cond=list(u=1),color="heat",zlim=c(-0.2,3.5))
par(op)

b <- gam(y~t2(v,w,u,k=c(25,5),d=c(2,1),bs=c("tp","cr"),full=TRUE),
         method="ML")
## more penalties now. numbers in labels like "r1" indicate which
## basis function of a null space is involved in the term.
pen.edf(b)
```

te

gamteteti

```
te(..., k=NA,bs="cr",m=NA,d=NA,by=NA,fx=FALSE,
     np=TRUE,xt=NULL,id=NULL,sp=NULL,pc=NULL)
ti(..., k=NA,bs="cr",m=NA,d=NA,by=NA,fx=FALSE,
     np=TRUE,xt=NULL,id=NULL,sp=NULL,mc=NULL,pc=NULL)

...      te(log(x),z)te(I(x/sd(x)),z)predict.gam
k        5^dchoose.k
bs       "cr""cs""cc""tp""ts"smooth.termssmooth.construct
m        mmp.splineDuchon.splineNA"cr"
d        d=c(2,1)
by       gam.modelsgam.modelslinear.functional.terms
fx       TRUEFALSE
np       TRUE
xt
id       id
sp       spgamgamm
mc       tiFALSETRUE'ti'
pc       NULLidentiifiabilityscasm
```

[tensor.prod.model.matrix](#)[tensor.prod.penalties](#)

$$f_1(x) + f_2(z) + f_3(x, z)$$

ti~ ti(x) + ti(z) + ti(x,z)steti

np=TRUE"cc""cr""cs"np=FALSE

[tensor.smooth.spec](#)[smooth.construct.tensor.smooth.spec](#)

margin [smooth.specs](#)

term

by by"NA"

fx TRUEFALSE

label

dim

mp TRUE

np TRUE

id idte

sp spte

inter TRUEtifALSE

mc mcti

<simon.wood@r-project.org>

[sgamgammsmooth.construct.tensor.smooth.spec](#)

```

# following shows how tensor prduct deals nicely with
# badly scaled covariates (range of x 5% of range of z )
require(mgcv)
test1 <- function(x,z,sx=0.3,sz=0.4) {
  x <- x*20
  (pi*sx*sz)*(1.2*exp(-(x-0.2)^2/sx^2-(z-0.3)^2/sz^2)+
  0.8*exp(-(x-0.7)^2/sx^2-(z-0.8)^2/sz^2))
}
n <- 500
old.par <- par(mfrow=c(2,2))
x <- runif(n)/20;z <- runif(n);
xs <- seq(0,1,length=30)/20;zs <- seq(0,1,length=30)
pr <- data.frame(x=rep(xs,30),z=rep(zs,rep(30,30)))
truth <- matrix(test1(pr$x,pr$z),30,30)
f <- test1(x,z)
y <- f + rnorm(n)*0.2
b1 <- gam(y~s(x,z))
persp(xs,zs,truth);title("truth")
vis.gam(b1);title("t.p.r.s")
b2 <- gam(y~te(x,z))
vis.gam(b2);title("tensor product")
b3 <- gam(y~ ti(x) + ti(z) + ti(x,z))
vis.gam(b3);title("tensor anova")

## now illustrate partial ANOVA decomp...
vis.gam(b3);title("full anova")
b4 <- gam(y~ ti(x) + ti(x,z,mc=c(0,1))) ## note z constrained!
vis.gam(b4);title("partial anova")
plot(b4)

par(old.par)

## now with a multivariate marginal....

test2<-function(u,v,w,sv=0.3,sw=0.4)
{ ((pi*sv*sw)*(1.2*exp(-(v-0.2)^2/sv^2-(w-0.3)^2/sw^2)+
  0.8*exp(-(v-0.7)^2/sv^2-(w-0.8)^2/sw^2)))*(u-0.5)^2*20
}
n <- 500
v <- runif(n);w<-runif(n);u<-runif(n)
f <- test2(u,v,w)
y <- f + rnorm(n)*0.2
# tensor product of 2D Duchon spline and 1D cr spline
m <- list(c(1,.5),rep(0,0)) ## example of list form of m
b <- gam(y~te(v,w,u,k=c(30,5),d=c(2,1),bs=c("ds","cr"),m=m))
plot(b)

```

tensor.prod.model.matrix

```

tensor.prod.model.matrix(X)
tensor.prod.penalties(S)
a%.%b

```

```

X          "matrix""dgCMatrix"
S
a          A
b          B

```

```

X[[1]]X[[2]]X[[m]]X[[1]][i,]%x%X[[2]][i,]%x% ... X[[m]][i,]%x%
A%.%B
S[[1]]S[[2]]S[[m]]I[[1]]I[[2]]I[[m]]
S[[1]]%x%I[[2]]%x% ... I[[m]]
I[[1]]%x%S[[2]]%x% ... I[[m]]

I[[1]]%x%I[[2]]%x% ... S[[m]]

```

<simon.wood@r-project.org>

[tesmooth.construct.tensor.smooth.spec](#)

```

require(mgcv)
## Dense row Kronecker product example...
X <- list(matrix(0:3,2,2),matrix(c(5:8,0,0),2,3))
tensor.prod.model.matrix(X)
X[[1]]%.%X[[2]]

## sparse equivalent...
Xs <- lapply(X,as,"dgCMatrix")
tensor.prod.model.matrix(Xs)
Xs[[1]]%.%Xs[[2]]

S <- list(matrix(c(2,1,1,2),2,2),matrix(c(2,1,0,1,2,1,0,1,2),3,3))
tensor.prod.penalties(S)
## Sparse equivalent...
Ss <- lapply(S,as,"dgCMatrix")
tensor.prod.penalties(Ss)

```

totalPenaltySpace

totalPenaltySpace(S, H, off, p)

S

H

off

p

$S[[i]] = B[[i]] \%*\% t(B[[i]])$

trichol

trichol(ld, sd)

ld

sd

$\text{dpttrfLAPACK} O(n) O(n^3)$

$\text{ldsdldsd} \mathbf{B} \mathbf{B}^T \mathbf{B} = \mathbf{T} \mathbf{T}$

<simon.wood@r-project.org>

bandchol

```
require(mgcv)
## simulate some diagonals...
set.seed(19); k <- 7
ld <- runif(k)+1
sd <- runif(k-1) -.5

## get diagonals of chol factor...
trichol(ld,sd)

## compare to dense matrix result...
A <- diag(ld);for (i in 1:(k-1)) A[i,i+1] <- A[i+1,i] <- sd[i]
R <- chol(A)
diag(R);diag(R[, -1])
```

trind.generator

```
trind.generator(K = 2, ifunc=FALSE, reverse= !ifunc)
```

K

ifunc TRUE

reverse ifunc==TRUE

```
m=1for(i in 1:K) for(j in i:K) for(k in j:K) for(l in k:K) {a[,m] <- something;
m <- m+1 }i4[i,j,k,l]mi3i2ifunc==TRUEi2i3i4i4(i,j,k,l)mK
```

```
i1i4ifunc==TRUEreverse==TRUEi1ri4r
```

```

library(mgcv)
A <- trind.generator(3,reverse=TRUE)

# All permutations of c(1, 2, 3) point to the same index (5)
A$i3[1, 2, 3]
A$i3[2, 1, 3]
A$i3[2, 3, 1]
A$i3[3, 1, 2]
A$i3[1, 3, 2]

## use reverse indices to pick out unique elements
## just for illustration...
A$i2;A$i2[A$i2r]
A$i3[A$i3r]

## same again using function indices...
A <- trind.generator(3,ifunc=TRUE)
A$i3(1, 2, 3)
A$i3(2, 1, 3)
A$i3(2, 3, 1)
A$i3(3, 1, 2)
A$i3(1, 3, 2)

```

Tweedie

```

gammgcvquasiTweedieptwppp
twgambamgammTweedie

```

```

Tweedie(p=1, link = power(0))
tw(theta = NULL, link = "log",a=1.01,b=1.99)

```

p	pp
link	"log""identity""inverse""sqrt"powerTweedie
theta	$p = (a + b \exp(\theta)) / (1 + \exp(\theta))$ pp
a	p
b	p

```

NNldTweedie
Tweediepoissontweediestatmodmgcvaic
ppquasitweedie

```

Tweediefamily

dvar mu

d2var mu

ls

twextended.family

p

<simon.wood@r-project.org>

[ldTweedierTweedie](#)

```
library(mgcv)
set.seed(3)
n<-400
## Simulate data...
dat <- gamSim(1,n=n,dist="poisson",scale=.2)
dat$y <- rTweedie(exp(dat$f),p=1.3,phi=.5) ## Tweedie response

## Fit a fixed p Tweedie, with wrong link ...
b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=Tweedie(1.25,power(.1)),
        data=dat)
plot(b,pages=1)
print(b)

## Same by approximate REML...
b1 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=Tweedie(1.25,power(.1)),
        data=dat,method="REML")
plot(b1,pages=1)
print(b1)

## estimate p as part of fitting

b2 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=tw(),
        data=dat,method="REML")
plot(b2,pages=1)
print(b2)

rm(dat)
```

twlss

[gam](#) $\phi\mu^p\phi p\mu$

`twlss(link=list("log","identity","identity"),a=1.01,b=1.99)`

`link`

`a`

`b`

$\mu \log \phi \mu^p \rho = \log(\phi) \theta p = \{a + b \exp(\theta)\} / \{1 + \exp(\theta)\}$

[NNldTweedie](#)

[ppquasi](#)tweedie

`general.family`

`<simon.wood@r-project.org>`

[Tweedie](#)[ldTweedie](#)[rTweedie](#)

```
library(mgcv)
set.seed(3)
n<-400
## Simulate data...
dat <- gamSim(1,n=n,dist="poisson",scale=.2)
dat$y <- rTweedie(exp(dat$f),p=1.3,phi=.5) ## Tweedie response

## Fit a fixed p Tweedie ...
b <- gam(list(y~s(x0)+s(x1)+s(x2)+s(x3),~1,~1),family=twlss(),
            data=dat)
plot(b,pages=1)
print(b)

eb <- exp(coef(b));nb <- length(eb)
eb[nb-1] ## scale
(1+2*eb[nb])/(1+eb[nb]) ## p

rm(dat)
```

uniquecombs

```
uniquecombs(x,ordered=FALSE)
```

```
x
ordered      TRUE
```

```
xuniquematchpaste0uniquematch
uniqueduplicated
x
```

```
x
"index"index[i]
```

```
as.characteras.character
*
```

```
<simon.wood@r-project.org>
```

```
uniqueduplicatedmatch
```

```
require(mgcv)

## matrix example...
X <- matrix(c(1,2,3,1,2,3,4,5,6,1,3,2,4,5,6,1,1,1),6,3,byrow=TRUE)
print(X)
Xu <- uniquecombs(X);Xu
ind <- attr(Xu,"index")
## find the value for row 3 of the original from Xu
Xu[ind[3,],];X[3,]

## same with fixed output ordering
Xu <- uniquecombs(X,TRUE);Xu
ind <- attr(Xu,"index")
## find the value for row 3 of the original from Xu
Xu[ind[3,],];X[3,]
```

```
## data frame example...
df <- data.frame(f=factor(c("er",3,"b","er",3,3,1,2,"b")),
  x=c(.5,1,1.4,.5,1,.6,4,3,1.7),
  bb = c(rep(TRUE,5),rep(FALSE,4)),
  fred = c("foo","a","b","foo","a","vf","er","r","g"),
  stringsAsFactors=FALSE)
uniquecombs(df)
```

`vcov.gam`

`gam`

```
## S3 method for class 'gam'
vcov(object, sandwich=FALSE, freq = FALSE, dispersion = NULL,unconditional=FALSE, ...)
```

```
object      gamgam()
sandwich
freq        TRUEFALSE
dispersion
unconditional TRUEfreq==FALSE
...
```

```
object$Vobject$Vpobject$VcgamObjectsandwich==TRUE
```

`freq`

`<simon.wood@r-project.org>`

`gam`

```
require(mgcv)
n <- 100
x <- runif(n)
y <- sin(x*2*pi) + rnorm(n)*.2
mod <- gam(y~s(x,bs="cc",k=10),knots=list(x=seq(0,1,length=10)))
diag(vcov(mod))
```

vis.gam

gamviewcond

```
vis.gam(x,view=NULL,cond=list(),n.grid=30,too.far=0,col=NA,  
        color="heat",contour.col=NULL,se=-1,type="link",  
        plot.type="persp",zlim=NULL,nCol=50,lp=1,...)
```

x	gamgam()
view	vis.gam
cond	view
n.grid	
too.far	viewtoo.farviewtoo.far
col	NAsecolorcolNA
color	se"topo""heat""cm""terrain""gray""bw""gray""bw"se
contour.col	plot.type="contour"NULL
se	sese
type	"link""response"
plot.type	"contour""persp"
zlim	NULL
nCol	
lp	
...	perspimagecontour ticktype="detailed"

viewsesetoo.far

...vis.gam

viewview

<simon.wood@r-project.org>

[perspgam](#)

```

library(mgcv)
set.seed(0)
n<-200;sig2<-4
x0 <- runif(n, 0, 1);x1 <- runif(n, 0, 1)
x2 <- runif(n, 0, 1)
y<-x0^2+x1*x2 +runif(n,-0.3,0.3)
g<-gam(y~s(x0,x1,x2))
old.par<-par(mfrow=c(2,2))
# display the prediction surface in x0, x1 ....
vis.gam(g,ticktype="detailed",color="heat",theta=-35)
vis.gam(g,se=2,theta=-35) # with twice standard error surfaces
vis.gam(g, view=c("x1","x2"),cond=list(x0=0.75)) # different view
vis.gam(g, view=c("x1","x2"),cond=list(x0=.75),theta=210,phi=40,
        too.far=.07)
# ..... areas where there is no data are not plotted

# contour examples...
vis.gam(g, view=c("x1","x2"),plot.type="contour",color="heat")
vis.gam(g, view=c("x1","x2"),plot.type="contour",color="terrain")
vis.gam(g, view=c("x1","x2"),plot.type="contour",color="topo")
vis.gam(g, view=c("x1","x2"),plot.type="contour",color="cm")

par(old.par)

# Examples with factor and "by" variables

fac<-rep(1:4,20)
x<-runif(80)
y<-fac+2*x^2+rnorm(80)*0.1
fac<-factor(fac)
b<-gam(y~fac+s(x))

vis.gam(b,theta=-35,color="heat") # factor example

z<-rnorm(80)*0.4
y<-as.numeric(fac)+3*x^2*z+rnorm(80)*0.1
b<-gam(y~fac+s(x,by=z))

vis.gam(b,theta=-35,color="heat",cond=list(z=1)) # by variable example

vis.gam(b,view=c("z","x"),theta=-135) # plot against by variable

```

XWXd

```

XWXd(X,w,k,ks,ts,dt,v,qc,nthreads=1,drop=NULL,ar.stop=-1,ar.row=-1,ar.w=-1,
      lt=NULL,rt=NULL)
XWyd(X,w,y,k,ks,ts,dt,v,qc,drop=NULL,ar.stop=-1,ar.row=-1,ar.w=-1,lt=NULL)

```



```

Xbd(X,beta,k,ks,ts,dt,v,qc,drop=NULL,lt=NULL)
diagXVXd(X,V,k,ks,ts,dt,v,qc,drop=NULL,nthreads=1,lt=NULL,rt=NULL)
ijXVXd(i,j,X,V,k,ks,ts,dt,v,qc,drop=NULL,nthreads=1,lt=NULL,rt=NULL)

```

```

X          ltrt"lpip"X"dgCMatrix""r""off"
w
y
beta
k
ks          ks[i,1]:(ks[i,2]-1)
ts          X
dt          X
v          v[[i]]qc[i]>0
qc          qc[i]>0
nthreads
drop
ar.stop      (ar.stop[i-1]+1):ar.stop[i]ar.rowar.w
ar.row
ar.w          ar.stop
lt          XXWXdNULLrt
rt          ltXNULLltltrtNULL
V
i           $XVX^T$ 
j

```

```

bamXWXdXTWXXWyXTWyXbdXβdiagXVXdXVXTijXVXdi,jij
"lpip"Xltrt
X"dgCMatrix"krkoffkroffXroffr[off[j]]:(off[j+1]-1)]jk

```

<simon.wood@r-project.org>

```

library(mgcv);library(Matrix)
## simulate some data creating a marginal matrix sequence...
set.seed(0);n <- 4000
dat <- gamSim(1,n=n,dist="normal",scale=2)
dat$x4 <- runif(n)
dat$y <- dat$y + 3*exp(dat$x4*15-5)/(1+exp(dat$x4*15-5))
dat$fac <- factor(sample(1:20,n,replace=TRUE))
G <- gam(y ~ te(x0,x2,k=5,bs="bs",m=1)+s(x1)+s(x4)+s(x3,fac,bs="fs"),
        fit=FALSE,data=dat,discrete=TRUE)
p <- ncol(G$X)
## create a sparse version...
Xs <- list(); r <- G$kd*0; off <- list()
for (i in 1:length(G$Xd)) Xs[[i]] <- as(G$Xd[[i]],"dgCMatrix")
for (j in 1:nrow(G$ks)) { ## create the reverse indices...
  nr <- nrow(Xs[[j]]) ## make sure we always tab to final stored row
  for (i in G$ks[j,1):(G$ks[j,2]-1)) {
    r[,i] <- (1:length(G$kd[,i]))[order(G$kd[,i])]
    off[[i]] <- cumsum(c(1,tabulate(G$kd[,i],nbins=nr)))-1
  }
}
attr(Xs,"off") <- off;attr(Xs,"r") <- r

par(mfrow=c(2,3))

beta <- runif(p)
Xb0 <- Xbd(G$Xd,beta,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
Xb1 <- Xbd(Xs,beta,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
range(Xb0-Xb1);plot(Xb0,Xb1,pch=".")

bb <- cbind(beta,beta+runif(p)*.3)
Xb0 <- Xbd(G$Xd,bb,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
Xb1 <- Xbd(Xs,bb,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
range(Xb0-Xb1);plot(Xb0,Xb1,pch=".")

p <- length(beta) ## extract full model matrix...
X <- matrix(Xbd(G$Xd,diag(p),G$kd,G$ks,G$ts,G$dt,G$v,G$qc),ncol=p)

w <- runif(n)
XWy0 <- XWyd(G$Xd,w,y=dat$y,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
XWy1 <- XWyd(Xs,w,y=dat$y,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
range(XWy1-XWy0);plot(XWy1,XWy0,pch=".")

yy <- cbind(dat$y,dat$y+runif(n)-.5)
XWy0 <- XWyd(G$Xd,w,y=yy,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
XWy1 <- XWyd(Xs,w,y=yy,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
range(XWy1-XWy0);plot(XWy1,XWy0,pch=".")

A <- XWXd(G$Xd,w,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
B <- XWXd(Xs,w,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
D <- crossprod(X,w*X) ## direct computation
range(A-D)
range(A-B);plot(A,B,pch=".")
## compute some cross product terms only...
A <- XWXd(G$Xd,w,G$kd,G$ks,G$ts,G$dt,G$v,G$qc,lt=1:3,rt=4:5)
range(A-D[1:nrow(A),(nrow(A)+1):ncol(D)])

```

```
V <- crossprod(matrix(runif(p*p),p,p))
ii <- c(20:30,100:200)
jj <- c(50:90,150:160)
V[ii,jj] <- 0;V[jj,ii] <- 0
d1 <- diagXVXd(G$Xd,V,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
Vs <- as(V,"dgCMatrix")
d2 <- diagXVXd(Xs,Vs,G$kd,G$ks,G$ts,G$dt,G$v,G$qc)
range(d1-d2);plot(d1,d2,pch=".")
```

ziP

[gambam](#)

```
ziP(theta = NULL, link = "identity",b=0)
```

```
theta       $\theta_1\theta_2$ 
link       "identity"
b
```

```
1 - p y > 0 p \mu^y / ((\exp(\mu) - 1) y!) \log \mu \eta = \log(-\log(1 - p)) \eta = \theta_1 + \{b + \exp(\theta_2)\} \log \mu
predicttype=="response" \theta_1 b + \exp(\theta_2)
```

extended.family

<simon.wood@r-project.org>

[ziplss](#)

```

rzip <- function(gamma,theta= c(-2,.3)) {
  ## generate zero inflated Poisson random variables, where
  ## lambda = exp(gamma), eta = theta[1] + exp(theta[2])*gamma
  ## and 1-p = exp(-exp(eta)).
  y <- gamma; n <- length(y)
  lambda <- exp(gamma)
  eta <- theta[1] + exp(theta[2])*gamma
  p <- 1- exp(-exp(eta))
  ind <- p > runif(n)
  y[!ind] <- 0
  np <- sum(ind)
  ## generate from zero truncated Poisson, given presence...
  y[ind] <- qpois(runif(np,dpois(0,lambda[ind]),1),lambda[ind])
  y
}

library(mgcv)
## Simulate some ziP data...
set.seed(1);n<-400
dat <- gamSim(1,n=n)
dat$y <- rzip(dat$f/4-1)

b <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=ziP(),data=dat)

b$outer.info ## check convergence!!
b
plot(b,pages=1)
plot(b,pages=1,unconditional=TRUE) ## add s.p. uncertainty
gam.check(b)
## more checking...
## 1. If the zero inflation rate becomes decoupled from the linear predictor,
## it is possible for the linear predictor to be almost unbounded in regions
## containing many zeroes. So examine if the range of predicted values
## is sane for the zero cases?
range(predict(b,type="response")[b$y==0])

## 2. Further plots...
par(mfrow=c(2,2))
plot(predict(b,type="response"),residuals(b))
plot(predict(b,type="response"),b$y);abline(0,1,col=2)
plot(b$linear.predictors,b$y)
qq.gam(b,rep=20,level=1)

## 3. Refit fixing the theta parameters at their estimated values, to check we
## get essentially the same fit...
thb <- b$family$getTheta()
b0 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=ziP(theta=thb),data=dat)
b;b0

## Example fit forcing minimum linkage of prob present and
## linear predictor. Can fix some identifiability problems.
b2 <- gam(y~s(x0)+s(x1)+s(x2)+s(x3),family=ziP(b=.3),data=dat)

```

ziplss

ziplssgam

```
ziplss(link=list("identity","identity"))
zipll(y,g,eta,deriv=0)
```

```
link
y
g
eta
deriv
```

```
ziplssgamgam
predict.gamtype"link""response"
```

```
s(x,m=1)Duchon.splines(x,z,m=c(1,.5))
zipllziplssziplss
```

ziplssgeneral.family

<simon.wood@r-project.org>

```
library(mgcv)
## simulate some data...
f0 <- function(x) 2 * sin(pi * x); f1 <- function(x) exp(2 * x)
f2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 *
      (10 * x)^3 * (1 - x)^10
n <- 500;set.seed(5)
x0 <- runif(n); x1 <- runif(n)
x2 <- runif(n); x3 <- runif(n)
```

```

## Simulate probability of potential presence...
eta1 <- f0(x0) + f1(x1) - 3
p <- binomial()$linkinv(eta1)
y <- as.numeric(runif(n)<p) ## 1 for presence, 0 for absence

## Simulate y given potentially present (not exactly model fitted!)...
ind <- y>0
eta2 <- f2(x2[ind])/3
y[ind] <- rpois(exp(eta2),exp(eta2))

## Fit ZIP model...
b <- gam(list(y~s(x2)+s(x3),~s(x0)+s(x1)),family=ziplss())
b$outer.info ## check convergence

summary(b)
plot(b,pages=1)

```


nlme

ACF

gls1me

ACF(object, maxLag, ...)

object

maxLag

...

<Bates@stat.wisc.edu>

[ACF.glsACF.lmeplot.ACF](#)

ACF.gls

glsform

```
## S3 method for class 'gls'
ACF(object, maxLag, resType, form, na.action, ...)
```

```
object          "gls"
maxLag
resType         "response""pearson""normalized""pearson"
form            ~ t~ t | gtgform~ 1
na.action       NAna.failACF.gls
...
```

lagACFACF

<bates@stat.wisc.edu>

[ACF.lmepplot.ACF](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
ACF(fm1, form = ~ 1 | Mare)

# Pinheiro and Bates, p. 255-257
fm1Dial.gls <- gls(rate ~
  (pressure+I(pressure^2)+I(pressure^3)+I(pressure^4))*QB,
  Dialyzer)

fm2Dial.gls <- update(fm1Dial.gls,
  weights = varPower(form = ~ pressure))

ACF(fm2Dial.gls, form = ~ 1 | Subject)
```

ACF.lme

lme

```
## S3 method for class 'lme'  
ACF(object, maxLag, resType, ...)
```

```
object          "lme"  
maxLag  
resType         "response""pearson""normalized""pearson"  
...
```

lagACFACF

<bates@stat.wisc.edu>

[ACF.glsplot.ACF](#)

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),  
          Ovary, random = ~ sin(2*pi*Time) | Mare)  
ACF(fm1, maxLag = 11)  
  
# Pinheiro and Bates, p240-241  
fm1Over.lme <- lme(follicles ~ sin(2*pi*Time) +  
                  cos(2*pi*Time), data=Ovary,  
                  random=pdDiag(~sin(2*pi*Time)) )  
(ACF.fm1Over <- ACF(fm1Over.lme, maxLag=10))  
plot(ACF.fm1Over, alpha=0.01)
```

Alfalfa

Alfalfa

CossackLadakRanger
NoneS1S2007
123456

x

allCoef

...mapdots

allCoef(..., extract)

...	extractcorStructvarFunc
extract	coef

extract...

[lmeStructnlmeStruct](#)

```
cs1 <- corAR1(0.1)
vf1 <- varPower(0.5)
allCoef(cs1, vf1)
```

anova.gls

```
TermsLNULLTermsNULLLNULLtest=TRUE
```

```
## S3 method for class 'gls'
anova(object, ..., test, type, adjustSigma, Terms, L, verbose)
```

```
object      "gls"
...         "gls""gnls""lm""lme""lmList""nlme""nlsList""nls"
test        object...TRUE
type        "sequential""marginal""sequential"
adjustSigma TRUEobject $\sqrt{n_{obs}/(n_{obs} - n_{par})}$ TRUE
Terms       NULL
L           NULL
verbose     TRUEverbose = FALSEFALSE
```

```
"anova.lme"
```

<bates@stat.wisc.edu>

```
glsgnlsnlmelmelogLik.glsAICBICprint.anova.lme
```

```
# AR(1) errors within each Mare
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
anova(fm1)
# variance changes with a power of the absolute fitted values?
fm2 <- update(fm1, weights = varPower())
anova(fm1, fm2)
```

```
# Pinheiro and Bates, p. 251-252
fm10rth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
  correlation = corSymm(form = ~ 1 | Subject),
  weights = varIdent(form = ~ 1 | age))
```

```

fm2Orth.gls <- update(fm1Orth.gls,
                      corr = corCompSymm(form = ~ 1 | Subject))
anova(fm1Orth.gls, fm2Orth.gls)

# Pinheiro and Bates, pp. 215-215, 255-260
#p. 215
fm1Dial.lme <-
  lme(rate ~ (pressure + I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
       Dialyzer, ~ pressure + I(pressure^2))
# p. 216
fm2Dial.lme <- update(fm1Dial.lme,
                      weights = varPower(form = ~ pressure))
# p. 255
fm1Dial.gls <- gls(rate ~ (pressure +
                        I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
                   Dialyzer)
fm2Dial.gls <- update(fm1Dial.gls,
                      weights = varPower(form = ~ pressure))
anova(fm1Dial.gls, fm2Dial.gls)
fm3Dial.gls <- update(fm2Dial.gls,
                      corr = corAR1(0.771, form = ~ 1 | Subject))
anova(fm2Dial.gls, fm3Dial.gls)
# anova.gls to compare a gls and an lme fit
anova(fm3Dial.gls, fm2Dial.lme, test = FALSE)

# Pinheiro and Bates, pp. 261-266
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
fm3Wheat2 <- update(fm1Wheat2,
                    corr = corRatio(c(12.5, 0.2),
                                     form = ~ latitude + longitude, nugget = TRUE))
# Test a specific contrast
anova(fm3Wheat2, L = c(-1, 0, 1))

```

anova.lme

TermsLNULLTermsNULLLNULLtest=TRUE

```

## S3 method for class 'lme'
anova(object, ..., test, type, adjustSigma, Terms, L, verbose)
## S3 method for class 'anova.lme'
print(x, verbose, ...)

```

object	"lme"
...	"gls""gnls""lm""lme""lmList""nlme""nlsList""nls"
test	object...TRUE
type	"sequential""marginal""sequential"
adjustSigma	TRUEobject $\sqrt{n_{obs}/(n_{obs} - n_{par})}$ TRUE

Terms	NULL
L	NULL
x	"anova.lme"
verbose	TRUE Everbose = FALSE FALSE

"anova.lme"

<bates@stat.wisc.edu>

[glsgnlslmelmelmeAICBICprint.anova.lmelogLik.lme](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
anova(fm1)
fm2 <- update(fm1, random = pdDiag(~age))
anova(fm1, fm2)

## Pinheiro and Bates, pp. 251-254 -----
fm1Orth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
  correlation = corSymm(form = ~ 1 | Subject),
  weights = varIdent(form = ~ 1 | age))
fm2Orth.gls <- update(fm1Orth.gls,
  corr = corCompSymm(form = ~ 1 | Subject))
## anova.gls examples:
anova(fm1Orth.gls, fm2Orth.gls)
fm3Orth.gls <- update(fm2Orth.gls, weights = NULL)
anova(fm2Orth.gls, fm3Orth.gls)
fm4Orth.gls <- update(fm3Orth.gls, weights = varIdent(form = ~ 1 | Sex))
anova(fm3Orth.gls, fm4Orth.gls)
# not in book but needed for the following command
fm3Orth.lme <- lme(distance ~ Sex*I(age-11), data = Orthodont,
  random = ~ I(age-11) | Subject,
  weights = varIdent(form = ~ 1 | Sex))
# Compare an "lme" object with a "gls" object (test would be non-sensical!)
anova(fm3Orth.lme, fm4Orth.gls, test = FALSE)

## Pinheiro and Bates, pp. 222-225 -----
op <- options(contrasts = c("contr.treatment", "contr.poly"))
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight, random = ~ Time)
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# Test a specific contrast
anova(fm2BW.lme, L = c("Time:Diet2" = 1, "Time:Diet3" = -1))

## Pinheiro and Bates, pp. 352-365 -----
```

```

fm1Theo.lis <- nlsList(
  conc ~ SSfol(Dose, Time, lKe, lKa, lCl), data=Theoph)
fm1Theo.lis
fm1Theo.nlme <- nlme(fm1Theo.lis)
fm2Theo.nlme <- update(fm1Theo.nlme, random= pdDiag(lKe+lKa+lCl~1) )
fm3Theo.nlme <- update(fm2Theo.nlme, random= pdDiag(    lKa+lCl~1) )

# Comparing the 3 nlme models
anova(fm1Theo.nlme, fm3Theo.nlme, fm2Theo.nlme)

options(op) # (set back to previous state)

```

as.matrix.corStruct

object

```

## S3 method for class 'corStruct'
as.matrix(x, ...)

```

```

x          "corStruct"
...

```

object

<bates@stat.wisc.edu>

[corClassescorMatrix](#)

```

cst1 <- corAR1(form = ~1|Subject)
cst1 <- Initialize(cst1, data = Orthodont)
as.matrix(cst1)

```

`as.matrix.pdMat`

`x`

```
## S3 method for class 'pdMat'  
as.matrix(x, ...)
```

```
x          "pdMat"  
...
```

`x`

`<bates@stat.wisc.edu>`

`pdMatcorMatrix`

```
as.matrix(pdSymm(diag(4)))
```

`as.matrix.reStruct`

`pdMatobject`

```
## S3 method for class 'reStruct'  
as.matrix(x, ...)
```

```
x          "reStruct"pdMat  
...
```

`object`

`<bates@stat.wisc.edu>`

[as.matrix.pdMatreStructpdMat](#)

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
as.matrix(rs1)
```

[asOneFormula](#)

...+

asOneFormula(..., omit)

...
omit

...omit

<bates@stat.wisc.edu>

[formulaall.vars](#)

```
asOneFormula(y ~ x + z | g, list(~ w, ~ t * sin(2 * pi)))
```

[Assay](#)

Assay

21
af
15

asTable

groupedData

asTable(object)

object groupedData

groupedDataobject

<bates@stat.wisc.edu>

[groupedData](#) is [Balanced](#) [balanced](#) [Grouped](#)

asTable(Orthodont)

```
# Pinheiro and Bates, p. 109
ergoStool.mat <- asTable(ergoStool)
```

augPred

primaryobjectgetGroups(object)NULLlevelmax(level)primary

```
augPred(object, primary, minimum, maximum, length.out, ...)
## S3 method for class 'lme'
augPred(object, primary = NULL,
         minimum = min(primary), maximum = max(primary),
         length.out = 51, level = Q, ...)
```

object	predict
primary	objectgetCovariateprimary
minimum	min(primary)
maximum	max(primary)
length.out	
level	
...	

object1originalpredictedpredict.groupVargroupVarfixed"augPred"

glslmelmList

<bates@stat.wisc.edu>

[plot.augPredgetGroupspredict](#)

```
fm1 <- lme(Orthodont, random = ~1)
augPred(fm1, length.out = 2, level = c(0,1))
```

balancedGrouped

groupedDatagroupedDatamatrixasTable

balancedGrouped(form, data, labels=NULL, units=NULL)

form	y ~ x g
data	dimnames
labels	xy
units	xy

groupedData

<bates@stat.wisc.edu>

groupedDataIsBalancedasTable

```
OrthoMat <- asTable( Orthodont )
Orth2 <- balancedGrouped(distance ~ age | Subject, data = OrthoMat,
  labels = list(x = "Age",
               y = "Distance from pituitary to pterygomaxillary fissure"),
  units = list(x = "(yr)", y = "(mm)"))
Orth2[ 1:10, ]      ## check the first few entries

# Pinheiro and Bates, p. 109
ergoStool.mat <- asTable(ergoStool)
balancedGrouped(effort~Type|Subject,
  data=ergoStool.mat)
```

bdf

bdf

bdf

<http://stat.gamma.rug.nl/snijders/multilevel.htm><http://www.stats.ox.ac.uk/~snijders/mlbook.htm>

```
summary(bdf)
```

```
## More examples, including lme() fits reproducing parts in the above
## book, are available in the R script files
system.file("mlbook", "ch04.R", package = "nlme") # and
system.file("mlbook", "ch05.R", package = "nlme")
```

BodyWeight

BodyWeight

23418567119101213151416

Cefamandole

Cefamandole

```
plot(Cefamandole)
fm1 <- nlsList(SSbiexp, data = Cefamandole)
summary(fm1)
```

Coef

```
"pdMat""corStruct""varFunc""reStruct""modelStruct"
coefficients<-coef<-
```

```
coef(object, ...) <- value
```

```
coefficients(object, ...) <- value
```

object	coef
...	
value	object

```
<bates@stat.wisc.edu>
```

[coef](#)

`coef.corStruct`

`object`

```
## S3 method for class 'corStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'corStruct'
coef(object, ...) <- value
```

```
object          "corStruct"
unconstrained   TRUEFALSETRUE
value           objectcoef{object}
...
```

`object``objectvalueObjectInitialize`

[corAR1](#)[corARMA](#)[corCAR1](#)[corCompSymm](#)[corExp](#)[corGaus](#)[corLinc](#)[corRatio](#)[corSpatial](#)[corSpher](#)
[corSymmInitialize](#)

```
cst1 <- corARMA(p = 1, q = 1)
coef(cst1)
```

coef.gnls

object

```
## S3 method for class 'gnls'
coef(object, ...)
```

```
object      "gnls"
...
```

object

<bates@stat.wisc.edu>

gnls

```
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
             weights = varPower())
coef(fm1)
```

coef.lme

ii

```
## S3 method for class 'lme'
coef(object, augFrame, level, data, which, FUN,
      omitGroupingFactor, subset, ...)
```

```
object      "lme"
augFrame     TRUEdataFALSEFALSE
level
data         augFrame = TRUEobject
which        datadata
FUN          dataFUNFUNorderedfactornumericmeanModedefactororderedModegsummary
            mode
omitGroupingFactor
            TRUEdataFALSE
subset
...
```



```
"coef.lme"level"ranef.lme""data.frame"
```

```
<bates@stat.wisc.edu>
```

[lmeranef.lmeplot.ranef.lmegsummary](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
coef(fm1)
coef(fm1, augFrame = TRUE)
```

```
coef.lmList
```

```
lmobjectlm
```

```
## S3 method for class 'lmList'
coef(object, augFrame, data, which, FUN,
      omitGroupingFactor, ...)
```

```
object      "lmList"lm
augFrame     TRUEobjectFALSEFALSE
data         augFrame = TRUEobject
which        object
FUN          FUNFUNorderedfactornumericmeanModefactororderedModegsummarymode
omitGroupingFactor
              TRUEdataFALSE
...
```

```
"coef.lmList""lm"object"lm""ranef.lmList""data.frame"
```

```
<bates@stat.wisc.edu>
```

[lmListfixed.effects.lmListranef.lmListplot.ranef.lmListgsummary](#)

```
fm1 <- lmList(distance ~ age|Subject, data = Orthodont)
coef(fm1)
coef(fm1, augFrame = TRUE)
```

`coef.modelStruct`

`modelStruct`

```
## S3 method for class 'modelStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'modelStruct'
coef(object, ...) <- value
```

```
object          "modelStruct""corStruct""varFunc"
unconstrained   TRUEFALSETRUE
value           objectcoef{object}
...
```

`object``objectvalueObjectInitialize``<bates@stat.wisc.edu>``Initialize`

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),
  corStruct = corAR1(0.3))
coef(lms1)
```

`coef.pdMat`

`object`

```
## S3 method for class 'pdMat'
coef(object, unconstrained, ...)
## S3 replacement method for class 'pdMat'
coef(object, ...) <- value
```

```
object      "pdMat"
unconstrained TRUEFALSEobjectTRUE
value       objectcoef{object}
...
```

object

objectvalue

pdMat

```
coef(pdSymm(diag(3)))
```

coef.reStruct

object

```
## S3 method for class 'reStruct'
coef(object, unconstrained, ...)
## S3 replacement method for class 'reStruct'
coef(object, ...) <- value
```

```
object      "reStruct"pdMat
unconstrained TRUEFALSETRUE
value       objectcoef(object)
...
```

object

objectvalue

<bates@stat.wisc.edu>

[coef.pdMatreStructpdMat](#)

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),
  B = pdDiag(2 * diag(4), form = ~Educ)))
coef(rs1)
```

[coef.varFunc](#)

object

```
## S3 method for class 'varFunc'
coef(object, unconstrained, allCoef, ...)
## S3 replacement method for class 'varIdent'
coef(object, ...) <- value
```

object	" varFunc "
unconstrained	TRUEFALSETRUE
allCoef	FALSETRUEFALSE
value	objectcoef{object}Object
...	

object

objectvalue

[varFunc](#)

```
vf1 <- varPower(1)
coef(vf1)
coef(vf1) <- 2
```

collapse

groupedData

collapse(object, ...)

object

...

<bates@stat.wisc.edu>

[collapse.groupedData](#)

collapse.groupedData

objectdisplayLevelpreservegroupedDatadisplayLevel

```
## S3 method for class 'groupedData'
collapse(object, collapseLevel, displayLevel,
         outer, inner, preserve, FUN, subset, ...)
```

object	groupedData
collapseLevel	
displayLevel	collapseLevel
outer	displayLevelTRUEdisplayLevelattr(object, "outer")NULL
inner	displayLevelTRUEattr(object, "outer")NULL
preserve	collapseLevelcollapseLevelNULL
FUN	objectcollapseLevelFUNFUNorderedfactornumericmeanModedefactor orderedModdegsummarymode
subset	NULl
...	

groupedDatadisplayLevelobjectcollapseLevel

<bates@stat.wisc.edu>

[groupedDataplot.nmGroupedData](#)

```
# collapsing by Dog
collapse(Pixel, collapseLevel = 1)
# same as collapse(Pixel, collapseLevel = "Dog")
```

compareFits

object1object2NA

compareFits(object1, object2, which)

object1object2 lmeImList
which object1object2

object1object2object1object1object2compareFits

<bates@stat.wisc.edu>

[plot.compareFitspairs.compareFitscomparePredcoefrandom.effects](#)

```
fm1 <- lmList(Orthodont)
fm2 <- lme(fm1)
(cF12 <- compareFits(coef(fm1), coef(fm2)))
```

`comparePred`

```
primaryobject1object2getGroups(object)NULLprimary
```

```
comparePred(object1, object2, primary, minimum, maximum,  
  length.out, level, ...)
```

```
object1object2 predict
```

```
primary      getCovariateprimary
```

```
minimum      min(primary)primarydataobject1
```

```
maximum      max(primary)primarydataobject1
```

```
length.out
```

```
level
```

```
...
```

```
object1originalcomparePredaugPred
```

```
glslmelmList
```

```
<bates@stat.wisc.edu>
```

[augPredgetGroups](#)

```
fm1 <- lme(distance ~ age * Sex, data = Orthodont, random = ~ age)  
fm2 <- update(fm1, distance ~ age)  
comparePred(fm1, fm2, length.out = 2)
```

corAR1

corAR1Initialize

corAR1(value, form, fixed)

value

form ~ t~ t | gtgform~ 1

fixed FALSE

corAR1

<bates@stat.wisc.edu>

[ACF.lmecorARMAcorClassesDim.corSpatialInitialize.corStructsummary.corStruct](#)

```
## covariate is observation order and grouping factor is Mare
cs1 <- corAR1(0.2, form = ~ 1 | Mare)
```

```
# Pinheiro and Bates, p. 236
cs1AR1 <- corAR1(0.8, form = ~ 1 | Subject)
cs1AR1. <- Initialize(cs1AR1, data = Orthodont)
corMatrix(cs1AR1.)
```

```
# Pinheiro and Bates, p. 240
fm1Ovar.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                  data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm2Ovar.lme <- update(fm1Ovar.lme, correlation = corAR1())
```

```
# Pinheiro and Bates, pp. 255-258: use in gls
fm1Dial.gls <-
  gls(rate ~(pressure + I(pressure^2) + I(pressure^3) + I(pressure^4))*QB,
       Dialyzer)
fm2Dial.gls <- update(fm1Dial.gls,
                     weights = varPower(form = ~ pressure))
fm3Dial.gls <- update(fm2Dial.gls,
                     corr = corAR1(0.771, form = ~ 1 | Subject))
```

```
# Pinheiro and Bates use in nlme:
```



```

# from p. 240 needed on p. 396
fm10var.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                  data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm50var.lme <- update(fm10var.lme,
                    correlation = corARMA(p = 1, q = 1))

# p. 396
fm10var.nlme <- nlme(follicles~
                    A+B*sin(2*pi*w*Time)+C*cos(2*pi*w*Time),
                    data=Ovary, fixed=A+B+C+w~1,
                    random=pdDiag(A+B+w~1),
                    start=c(fixef(fm50var.lme), 1) )
# p. 397
fm20var.nlme <- update(fm10var.nlme,
                    correlation=corAR1(0.311) )

```

corARMA

corARMAInitialize

corARMA(value, form, p, q, fixed)

value	p + q
form	~ t~ t gtgform~ 1
pq	
fixed	FALSE

corARMA

<bates@stat.wisc.edu>

[corAR1](#)[corClassesInitialize](#)[corStructsummary](#)[corStruct](#)

```
## ARMA(1,2) structure, with observation order as a covariate and
## Mare as grouping factor
cs1 <- corARMA(c(0.2, 0.3, -0.1), form = ~ 1 | Mare, p = 1, q = 2)

# Pinheiro and Bates, p. 237
cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
corMatrix(cs1ARMA)

cs2ARMA <- corARMA(c(0.8, 0.4), form = ~ 1 | Subject, p=1, q=1)
cs2ARMA <- Initialize(cs2ARMA, data = Orthodont)
corMatrix(cs2ARMA)

# Pinheiro and Bates use in nlme:
# from p. 240 needed on p. 396
fm10var.lme <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time),
                  data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm50var.lme <- update(fm10var.lme,
                    correlation = corARMA(p = 1, q = 1))
# p. 396
fm10var.nlme <- nlme(follicles~
  A+B*sin(2*pi*w*Time)+C*cos(2*pi*w*Time),
  data=Ovary, fixed=A+B+C+w~1,
  random=pdDiag(A+B+w~1),
  start=c(fixef(fm50var.lme), 1) )
# p. 397
fm30var.nlme <- update(fm10var.nlme,
  correlation=corARMA(p=0, q=2) )
```

corCAR1

corCAR1Initialize

corCAR1(value, form, fixed)

value

form ~ t~ t | gtgform~ 1

fixed FALSE

corCAR1

<bates@stat.wisc.edu>

[corClassesInitialize.corStructsummary.corStruct](#)

```
## covariate is Time and grouping factor is Mare
cs1 <- corCAR1(0.2, form = ~ Time | Mare)

# Pinheiro and Bates, pp. 240, 243
fm10var.lme <- lme(follicles ~
  sin(2*pi*Time) + cos(2*pi*Time),
  data = Ovary, random = pdDiag(~sin(2*pi*Time)))
fm40var.lme <- update(fm10var.lme,
  correlation = corCAR1(form = ~Time))
```

corClasses

corStruct

corAR1
corARMA
corCAR1
corCompSymm
corExp
corGaus
corLin
corRatio
corSpher
corSymm

corStruct[corMatrixcoef](#)corSymmcorAR1

<bates@stat.wisc.edu>

[corAR1corARMAcorCAR1corCompSymmcorExp](#)
[corGauscorLincorRatiocorSpher](#)
[corSymmsummary.corStruct](#)

corCompSymm

corCompSymmInitialize

corCompSymm(value, form, fixed)

value

form ~ t~ t | gtgform~ 1

fixed FALSE

corCompSymm

<bates@stat.wisc.edu>

[corClassesInitialize.corStructsummary.corStruct](#)

```
## covariate is observation order and grouping factor is Subject
cs1 <- corCompSymm(0.5, form = ~ 1 | Subject)
cs1 # Uninitialized ...
```

```
# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs2CompSymm <- corCompSymm(value = 0.3, form = ~ age | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)
```

corExp

"corExp" *dnr* $\exp(-r/d)(1 - n) \exp(-r/d)$ Initialize

corExp(value, form, nugget, metric, fixed)

value	nuggetFALSEvaluenuggetTRUEvaluenumeric(0)object
form	$\sim S1 + \dots + Sp \sim S1 + \dots + Sp \mid gS1Sp$ $gform \sim 1$
nugget	FALSE
metric	"euclidean" "maximum" "manhattan" "euclidean"
fixed	FALSE

"corExp" "corSpatial"

<bates@stat.wisc.edu>

[corClassesInitialize.corStructsummary.corStructdist](#)

```
sp1 <- corExp(form = ~ x + y + z)

# Pinheiro and Bates, p. 238
spatDat <- data.frame(x = (0:4)/4, y = (0:4)/4)

cs1Exp <- corExp(1, form = ~ x + y)
cs1Exp <- Initialize(cs1Exp, spatDat)
corMatrix(cs1Exp)

cs2Exp <- corExp(1, form = ~ x + y, metric = "man")
cs2Exp <- Initialize(cs2Exp, spatDat)
corMatrix(cs2Exp)

cs3Exp <- corExp(c(1, 0.2), form = ~ x + y,
                 nugget = TRUE)
cs3Exp <- Initialize(cs3Exp, spatDat)
corMatrix(cs3Exp)
```

```
# example lme(..., corExp ...)
# Pinheiro and Bates, pp. 222-247
# p. 222
options(contrasts = c("contr.treatment", "contr.poly"))
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                random = ~ Time)

# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p. 246
fm3BW.lme <- update(fm2BW.lme,
                  correlation = corExp(form = ~ Time))

# p. 247
fm4BW.lme <-
  update(fm3BW.lme, correlation = corExp(form = ~ Time,
                                         nugget = TRUE))
anova(fm3BW.lme, fm4BW.lme)
```

corFactor

"corStruct"

corFactor(object, ...)

object

...

<bates@stat.wisc.edu>

[corFactor.corStruct](#)[recalc.corStruct](#)

`corFactor.corStruct`

`object` $\Sigma\Sigma L\Sigma = L'LL^{-t}$

S3 method for class 'corStruct'
`corFactor(object, ...)`

`object` `"corStruct"`Initialize
...

`object`

`corMatrix`

`<bates@stat.wisc.edu>`

`corFactorcorMatrix.corStructrecalc.corStructInitialize.corStruct`

```
cs1 <- corAR1(form = ~1 | Subject)
cs1 <- Initialize(cs1, data = Orthodont)
corFactor(cs1)
```

`corGaus`

`corGaus` $dn\exp(-(r/d)^2)(1-n)\exp(-(r/d)^2)$ Initialize

`corGaus(value, form, nugget, metric, fixed)`

<code>value</code>	<code>nuggetFALSE</code> <code>value</code> <code>nuggetTRUE</code> <code>value</code> <code>numeric(0)</code> <code>object</code>
<code>form</code>	<code>~ S1+...+Sp</code> <code>~ S1+...+Sp gS1Sp</code> <code>gform</code> <code>~ 1</code>
<code>nugget</code>	<code>FALSE</code>
<code>metric</code>	<code>"euclidean"</code> <code>"maximum"</code> <code>"manhattan"</code> <code>"euclidean"</code>
<code>fixed</code>	<code>FALSE</code>

corGauscorSpatial

<bates@stat.wisc.edu>

[Initialize.corStructsummary.corStructdist](#)

```
sp1 <- corGaus(form = ~ x + y + z)

# example lme(..., corGaus ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                 random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
                   correlation = corExp(form = ~ Time))
# p. 249
fm8BW.lme <- update(fm3BW.lme, correlation = corGaus(form = ~ Time))
```

corLin

$\text{corLindnr} < d1 - (r/d)(1 - n)(1 - (r/d))r \geq d\text{Initialize}$

corLin(value, form, nugget, metric, fixed)

value	nuggetFALSEvaluenuggetTRUEvaluenumeric(0)object
form	~ S1+...+Sp~ S1+...+Sp gS1Spgform~ 1
nugget	FALSE
metric	"euclidean""maximum""manhattan""euclidean"
fixed	FALSE

corLincorSpatial

<bates@stat.wisc.edu>

[Initialize.corStructsummary.corStructdist](#)

```
sp1 <- corLin(form = ~ x + y)

# example lme(..., corLin ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                 random = ~ Time)
# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
                   correlation = corExp(form = ~ Time))
# p. 249
fm7BW.lme <- update(fm3BW.lme, correlation = corLin(form = ~ Time))
```

corMatrix

"corStruct"

corMatrix(object, ...)

object

...

<bates@stat.wisc.edu>

[corMatrix.corStructcorMatrix.pdMatcorMatrix.reStruct](#)

corMatrix.corStruct

covariateobject $\Sigma\Sigma L\Sigma = L' L \text{corr} = \text{FALSE} L^{-t}$

```
## S3 method for class 'corStruct'
corMatrix(object, covariate, corr, ...)
```

```
object          "corStruct"
covariate        getCovariate(object)
corr             TRUEobjectFALSE
...
```

covariatecovariatecovariate

<bates@stat.wisc.edu>

corFactor.corStructInitialize.corStruct

```
cs1 <- corAR1(0.3)
corMatrix(cs1, covariate = 1:4)
corMatrix(cs1, covariate = 1:4, corr = FALSE)

# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)

# Pinheiro and Bates, p. 226
cs1Symm <- corSymm(value = c(0.2, 0.1, -0.1, 0, 0.2, 0),
                  form = ~ 1 | Subject)
cs1Symm <- Initialize(cs1Symm, data = Orthodont)
corMatrix(cs1Symm)

# Pinheiro and Bates, p. 236
cs1AR1 <- corAR1(0.8, form = ~ 1 | Subject)
cs1AR1 <- Initialize(cs1AR1, data = Orthodont)
corMatrix(cs1AR1)

# Pinheiro and Bates, p. 237
cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
```

```
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
corMatrix(cs1ARMA)

# Pinheiro and Bates, p. 238
spatDat <- data.frame(x = (0:4)/4, y = (0:4)/4)
cs1Exp <- corExp(1, form = ~ x + y)
cs1Exp <- Initialize(cs1Exp, spatDat)
corMatrix(cs1Exp)
```

corMatrix.pdMat

object

```
## S3 method for class 'pdMat'
corMatrix(object, ...)
```

object **"pdMat"**
...

object

<bates@stat.wisc.edu>

as.matrix.pdMatpdMatrix

```
pd1 <- pdSymm(diag(1:4))
corMatrix(pd1)
```

corMatrix.reStruct

pdMatobject

```
## S3 method for class 'reStruct'
corMatrix(object, ...)
```

object **"reStruct"**pdMat
...

object

<bates@stat.wisc.edu>

[as.matrix.reStruct](#)[corMatrixreStruct](#)[pdMat](#)

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
corMatrix(rs1)
```

corNatural

corNatural[pdNaturalInitialize](#)

corNatural(value, form, fixed)

value	numeric(0)object
form	~ t~ t gtgform~ 1
fixed	FALSE

corNatural

<bates@stat.wisc.edu>

[Initialize.corNatural](#)[pdNaturalsummary.corNatural](#)

```
## covariate is observation order and grouping factor is Subject
cs1 <- corNatural(form = ~ 1 | Subject)
cs1 # Uninitialized ...
summary(Initialize(cs1, data = Orthodont))
```

corRatio

$\text{corRatio} = \frac{nr}{1 + (r/d)^2} \frac{1 - n}{1 + (r/d)^2}$ Initialize

corRatio(value, form, nugget, metric, fixed)

value	nuggetFALSEvaluenuggetTRUEvaluenumeric(0)object
form	~ S1+...+Sp~ S1+...+Sp gS1Spgform~ 1
nugget	FALSE
metric	"euclidean""maximum""manhattan""euclidean"
fixed	FALSE

corRatiocorSpatial

<bates@stat.wisc.edu>

[Initialize.corStructsummary.corStructdist](#)

```
sp1 <- corRatio(form = ~ x + y + z)

# example lme(..., corRatio ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
  random = ~ Time)

# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
  correlation = corExp(form = ~ Time))
# p. 249
fm5BW.lme <- update(fm3BW.lme, correlation =
  corRatio(form = ~ Time))

# example gls(..., corRatio ...)
# Pinheiro and Bates, pp. 261, 263
```

```
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 263
fm3Wheat2 <- update(fm1Wheat2, corr =
  corRatio(c(12.5, 0.2),
    form = ~ latitude + longitude,
    nugget = TRUE))
```

corSpatial

corSpatial corExp corGaus corLincorRatio corSpher type corSpatialInitialize

corSpatial(value, form, nugget, type, metric, fixed)

value	nugget FALSE evaluenugget TRUE evaluenumeric(0) object
form	~ S1+...+Sp ~ S1+...+Sp gS1Sp gform ~ 1
nugget	FALSE
type	"spherical" "exponential" "gaussian" "linear" "rational" corSpher corExp corGaus corLincorRatio "spherical"
metric	"euclidean" "maximum" "manhattan" "euclidean"
fixed	FALSE

type corSpatial

<bates@stat.wisc.edu>

[corExp](#) [corGaus](#) [corLincorRatio](#) [corSpherInitialize](#) [corStructsummary](#) [corStructdist](#)

```
sp1 <- corSpatial(form = ~ x + y + z, type = "g", metric = "man")
```

corSpher

$\text{corSpher}dnr < d1 - 1.5(r/d) + 0.5(r/d)^3(1 - n)(1 - 1.5(r/d) + 0.5(r/d)^3)r \geq d\text{Initialize}$

corSpher(value, form, nugget, metric, fixed)

value	nuggetFALSEvaluenuggetTRUEvaluenumeric(0)object
form	~ S1+...+Sp~ S1+...+Sp gS1Spgform~ 1
nugget	FALSE
metric	"euclidean""maximum""manhattan""euclidean"
fixed	FALSE

corSphercorSpatial

<bates@stat.wisc.edu>

[Initialize.corStructsummary.corStructdist](#)

```
sp1 <- corSpher(form = ~ x + y)

# example lme(..., corSpher ...)
# Pinheiro and Bates, pp. 222-249
fm1BW.lme <- lme(weight ~ Time * Diet, BodyWeight,
                 random = ~ Time)

# p. 223
fm2BW.lme <- update(fm1BW.lme, weights = varPower())
# p 246
fm3BW.lme <- update(fm2BW.lme,
                   correlation = corExp(form = ~ Time))
# p. 249
fm6BW.lme <- update(fm3BW.lme,
                   correlation = corSpher(form = ~ Time))

# example gls(..., corSpher ...)
# Pinheiro and Bates, pp. 261, 263
```

```
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 262
fm2Wheat2 <- update(fm1Wheat2, corr =
  corSpher(c(28, 0.2),
    form = ~ latitude + longitude, nugget = TRUE))
```

corSymm

corSymmInitialize

corSymm(value, form, fixed)

value	numeric(0) object
form	~ t~ t gtgform~ 1
fixed	FALSE

corSymm

<bates@stat.wisc.edu>

[Initialize.corSymmsummary.corSymm](#)

```
## covariate is observation order and grouping factor is Subject
cs1 <- corSymm(form = ~ 1 | Subject)
```

```
# Pinheiro and Bates, p. 225
cs1CompSymm <- corCompSymm(value = 0.3, form = ~ 1 | Subject)
cs1CompSymm <- Initialize(cs1CompSymm, data = Orthodont)
corMatrix(cs1CompSymm)
```

```
# Pinheiro and Bates, p. 226
cs1Symm <- corSymm(value =
  c(0.2, 0.1, -0.1, 0, 0.2, 0),
  form = ~ 1 | Subject)
cs1Symm <- Initialize(cs1Symm, data = Orthodont)
corMatrix(cs1Symm)
```

```
# example gls(..., corSpher ...)
```



```
# Pinheiro and Bates, pp. 261, 263
fm1Wheat2 <- gls(yield ~ variety - 1, Wheat2)
# p. 262
fm2Wheat2 <- update(fm1Wheat2, corr =
  corSpher(c(28, 0.2),
    form = ~ latitude + longitude, nugget = TRUE))

# example gls(..., corSymm ... )
# Pinheiro and Bates, p. 251
fm1Orth.gls <- gls(distance ~ Sex * I(age - 11), Orthodont,
  correlation = corSymm(form = ~ 1 | Subject),
  weights = varIdent(form = ~ 1 | age))
```

Covariate

"varFunc"

```
covariate(object) <- value
```

object	covariate
value	object

<bates@stat.wisc.edu>

[covariate<-varFuncgetCovariate](#)

Covariate.varFunc

```
objectvalueobjectvaluegetCovariate(object)
```

```
## S3 replacement method for class 'varFunc'
covariate(object) <- value
```

object	"varFunc"
value	object

varFuncobjectcovariatevalue

<bates@stat.wisc.edu>

[getCovariate.varFunc](#)

```
vf1 <- varPower(1.1, form = ~age)
covariate(vf1) <- Orthodont[["age"]]
```

Dialyzer

Dialyzer

1082635971417201112161314181519

200300

Dim

"[corSpatial](#)" "[corStruct](#)" "pdCompSymm" "pdDiag" "pdIdent" "pdMat" "pdSymm"

Dim(object, ...)

object

...

dim

<bates@stat.wisc.edu>

[Dim.corSpatial](#)[Dim.pdMat](#)[Dim.corStruct](#)

Dim.corSpatial

groupsDimobject

S3 method for class 'corSpatial'
Dim(object, groups, ...)

object "[corSpatial](#)"

groups

...

N groups

M

spClass corSphercorExpcorGauscorLin

sumLenSq

len

start

<bates@stat.wisc.edu>

DimDim.corStruct

```
Dim(corGaus(), getGroups(Orthodont))
```

```
cs1ARMA <- corARMA(0.4, form = ~ 1 | Subject, q = 1)
cs1ARMA <- Initialize(cs1ARMA, data = Orthodont)
Dim(cs1ARMA)
```

Dim.corStruct

groupsDimobject

```
## S3 method for class 'corStruct'
Dim(object, groups, ...)
```

```
object          "corStruct"
groups
...
```

```
N              groups
M
maxLen
sumLenSq
len
start
```

<bates@stat.wisc.edu>

DimDim.corSpatial

```
Dim(corAR1(), getGroups(Orthodont))
```

Dim.pdMat

object

```
## S3 method for class 'pdMat'  
Dim(object, ...)
```

```
object      "pdMat"  
...
```

object

<bates@stat.wisc.edu>

Dim

```
Dim(pdSymm(diag(3)))
```

Earthquake

Earthquake

2016141038232261372118154121959121711

01

ergoStool

ergoStool

T1T2T3T4

```
fm1 <-  
  lme(effort ~ Type, data = ergoStool, random = ~ 1 | Subject)  
anova( fm1 )
```

Fatigue

Fatigue

123456789101112131415161718192021

`fdHess`

```
fdHess(pars, fun, ...,
       .relStep = .Machine$double.eps^(1/3), minAbsPar = 0)
```

```
pars      fun
fun       pars
...       fun
.relStep  .Machine$double.eps
minAbsPar
```

```
mean      funpars
gradient  length(pars)
Hessian
```

```
<bates@stat.wisc.edu>
```

```
(fdH <- fdHess(c(12.3, 2.34), function(x) x[1]*(1-exp(-0.4*x[2]))))
stopifnot(length(fdH$ mean) == 1,
          length(fdH$ gradient) == 2,
          identical(dim(fdH$ Hessian), c(2L, 2L)))
```

`fitted.glsStruct`

```
object
```

```
## S3 method for class 'glStruct'
fitted(object, glsFit, ...)
```

```
object      "glsStruct"corStruct"varFunc"  
glsFit      logLikbetasigmavarBetafittedresidualsattr(object, "glsFit")  
...
```

```
object
```

```
glsfitted.gls
```

```
<bates@stat.wisc.edu>
```

```
glsresiduals.glsStruct
```

```
fitted.gnlsStruct
```

```
object
```

```
## S3 method for class 'gnlsStruct'  
fitted(object, ...)
```

```
object      "gnlsStruct"corStructvarFunc  
...
```

```
object
```

```
gnlsfitted.gnls
```

```
<bates@stat.wisc.edu>
```

```
gnlsresiduals.gnlsStruct
```

fitted.lme

iii

```
## S3 method for class 'lme'  
fitted(object, level, asList, ...)
```

```
object      "lme"  
level       object  
asList      TRUElevellevelFALSE  
...
```

```
levelasList = TRUEasList = FALSElevelnapredict
```

```
<bates@stat.wisc.edu>
```

[lmeresiduals.lme](#)

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)  
fitted(fm1, level = 0:1)
```

fitted.lmeStruct

iii

```
## S3 method for class 'lmeStruct'  
fitted(object, level, conLin, lmeFit, ...)
```

```
object      "lmeStruct"reStructcorStructvarFunc  
level       object  
conLin      "Xy"Xy"logLik"attr(object, "conLin")  
lmeFit      betabattr(object, "lmeFit")  
...
```

levellevel

lmefitted.lme

<bates@stat.wisc.edu>

lmefitted.lmeresiduals.lmeStruct

fitted.lmList

lmobjectobject

```
## S3 method for class 'lmList'
fitted(object, subset, asList, ...)
```

```
object      "lmList"lm
subset      lmobjectNULL
asList      TRUEFALSE
...
```

lmobjectlmobject

<bates@stat.wisc.edu>

lmListresiduals.lmList

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
fitted(fm1)
```

fitted.nlmeStruct

iii

```
## S3 method for class 'nlmeStruct'  
fitted(object, level, conLin, ...)
```

```
object      "nlmeStruct"reStructcorStructvarFunc  
level       object  
conLin      "Xy"Xy"logLik"attr(object, "conLin")  
...
```

```
levellevel
```

```
nlmefitted.nlme
```

```
<bates@stat.wisc.edu>
```

[nlmeresiduals.nlmeStruct](#)

fixed.effects

```
lmListlme  
fixed.effectsfixef
```

```
fixed.effects(object, ...)  
fixef(object, ...)
```

```
object  
...
```

[fixef.lmList](#)

`fixef.lmList`

`lmobject`

```
## S3 method for class 'lmList'
fixef(object, ...)
```

```
object          "lmList"lm
...
```

`lmobject``<bates@stat.wisc.edu>``lmListrandom.effects.lmList`

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
fixef(fm1)
fixed.effects(fm1) # the same, using the longer alias
```

`formula.pdBlocked`

`formulapdMatxasList=TRUEasList=FALSEpdMatformulaNULL`

```
## S3 method for class 'pdBlocked'
formula(x, asList, ...)
```

```
x          "pdBlocked"
asList      TRUExFALSEFALSE
...
```

`NULL``<bates@stat.wisc.edu>``pdBlockedpdMat`

```
pd1 <- pdBlocked(list(~ age, ~ Sex - 1))
formula(pd1)
formula(pd1, asList = TRUE)
```

formula.pdMat

pdMat

```
## S3 method for class 'pdMat'  
formula(x, asList, ...)
```

```
x          "pdMat"  
asList     asList  
...
```

xformulaNULL

formula(x)pdMat

<bates@stat.wisc.edu>

pdMat

```
pd1 <- pdSymm(~Sex*age)  
formula(pd1)
```

formula.reStruct

x

```
## S3 method for class 'reStruct'  
formula(x, asList, ...)
```

```
x          "reStruct"pdMat  
asList     asList  
...
```

x

<bates@stat.wisc.edu>

[formula](#)

```
rs1 <- reStruct(list(A = pdDiag(diag(2), ~age), B = ~1))
formula(rs1)
```

gapply

groups

```
gapply(object, which, FUN, form, level, groups, ...)
```

object	groupedDatadata.frame"data.frame"
which	objectFUNobject
FUN	objectgroups
form	objectformula(object)
level	
groups	getGroups(object, form, level)
...	FUNna.rm = TRUE

groups

<bates@stat.wisc.edu>

[gsummary](#)

```
## Find number of non-missing "conc" observations for each Subject
gapply( Phenobarb, FUN = function(x) sum(!is.na(x$conc)) )
```

```
# Pinheiro and Bates, p. 127
table( gapply(Quinidine, "conc", function(x) sum(!is.na(x))) )
changeRecords <- gapply( Quinidine, FUN = function(frm)
  any(is.na(frm[["conc"]]) & is.na(frm[["dose"]])) )
```

Gasoline

Gasoline

(lbf/in²)

APIvaporASTMendpoint

getCovariate

corStructcorSpatialdata.framevarFunc

getCovariate(object, form, data)

object	covariate
form	formula(object)
data	form

<bates@stat.wisc.edu>

[getCovariate.corStruct](#)[getCovariate.data.frame](#)[getCovariate.varFunc](#)
[getCovariateFormula](#)

`getCovariate.corStruct`

`object`

```
## S3 method for class 'corStruct'
getCovariate(object, form, data)
```

<code>object</code>	<code>corStruct</code>
<code>form</code>	<code>formula(object)</code>
<code>data</code>	<code>formobject</code>

`object``<bates@stat.wisc.edu>`

[getCovariate](#)

```
cs1 <- corAR1(form = ~ 1 | Subject)
getCovariate(cs1, data = Orthodont)
```

`getCovariate.data.frame`

`form|object`

```
## S3 method for class 'data.frame'
getCovariate(object, form, data)
```

<code>object</code>	<code>data.frame</code>
<code>form</code>	<code>objectformula(object)</code>
<code>data</code>	

formobject

<bates@stat.wisc.edu>

[getCovariateFormula](#)

getCovariate(Orthodont)

getCovariate.varFunc

object

```
## S3 method for class 'varFunc'  
getCovariate(object, form, data)
```

object	varFunc
form	objectformula(object)
data	data

objectcovariateNULL

<bates@stat.wisc.edu>

[covariate<- .varFunc](#)

```
vf1 <- varPower(1.1, form = ~age)  
covariate(vf1) <- Orthodont[["age"]]  
getCovariate(vf1)
```

getCovariateFormula

formula(object)|

getCovariateFormula(object)

object

formula(object)

<bates@stat.wisc.edu>

[getCovariate](#)

getCovariateFormula(y ~ x | g)
getCovariateFormula(y ~ x)

getData

glslmelmList

getData(object)

object

<bates@stat.wisc.edu>

[getData.glsl](#)[getData.lme](#)[getData.lmList](#)

getData.gls

object

```
## S3 method for class 'gls'
getData(object)
```

object gls

dataobjectna.actionssubsetobjectNULL

<bates@stat.wisc.edu>

[glsgetData](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), data = Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getData(fm1)
```

getData.lme

object

```
## S3 method for class 'lme'
getData(object)
```

object lme

dataobjectna.actionssubsetobjectNULL
na.action = na.omitna.action = na.exclude

<bates@stat.wisc.edu>

[lmegetData](#)

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), data = Ovary,
           random = ~ sin(2*pi*Time))
getData(fm1)
```

`getData.lmList`

`object`

```
## S3 method for class 'lmList'
getData(object)
```

`object` `lmListlm`

`dataobjectna.actionssubsetobjectNULL`

`<bates@stat.wisc.edu>`

[lmListgetData](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
getData(fm1)
```

`getGroups`

`corStructdata.frameglslme1lmListvarFunc`

`getGroups(object, form, level, data, sep)`

`object`

`form` `|formula(object)`

`level`

`data` `form`

`sep` `'/'`

`<bates@stat.wisc.edu>`

[getGroupsFormula](#)

[getGroups.corStructgetGroups.data.framegetGroups.glsgetGroups.lmList](#)

[getGroups.lmegetGroups.varFunc](#)

`getGroups.corStruct`

`object`

```
## S3 method for class 'corStruct'
getGroups(object, form, level, data, sep)
```

<code>object</code>	<code>corStruct</code>
<code>form</code>	<code>formula(object)</code>
<code>level</code>	
<code>data</code>	<code>formobject</code>
<code>sep</code>	<code>'/'</code>

`objectNULL``<bates@stat.wisc.edu>``getGroups`

```
cs1 <- corAR1(form = ~ 1 | Subject)
getGroups(cs1, data = Orthodont)
```

`getGroups.data.frame`

`|formobjectlevelformlevel > 1level`

```
## S3 method for class 'data.frame'
getGroups(object, form, level, data, sep)
```

<code>object</code>	<code>data.frame</code>
<code>form</code>	<code> formula(object)</code>
<code>level</code>	
<code>data</code>	
<code>sep</code>	<code>'/'</code>

level

<bates@stat.wisc.edu>

[getGroupsFormula](#)

```
getGroups(Pixel)
getGroups(Pixel, level = 2)
```

getGroups.gls

object

```
## S3 method for class 'gls'
getGroups(object, form, level, data, sep)
```

object	gls
form	formula(object)
level	
data	form
sep	'/'

objectcorStructNULL

<bates@stat.wisc.edu>

[glscorClasses](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getGroups(fm1)
```

`getGroups.lme`

`objectlevel`

```
## S3 method for class 'lme'
getGroups(object, form, level, data, sep)
```

```
object      lme
form
level       object
data
sep         '/'
```

`level``<bates@stat.wisc.edu>``lme`

```
fm1 <- lme(pixel ~ day + day^2, Pixel,
  random = list(Dog = ~day, Side = ~1))
getGroups(fm1, level = 1:2)
```

`getGroups.lmList`

`lmobject`

```
## S3 method for class 'lmList'
getGroups(object, form, level, data, sep)
```

```
object      lmListlm
form        |formula(object)
level
data        form
sep         '/'
```

lmobject

<bates@stat.wisc.edu>

lmList

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
getGroups(fm1)
```

getGroups.varFunc

object

```
## S3 method for class 'varFunc'
getGroups(object, form, level, data, sep)
```

object	varFunc
form	formula(object)
level	
data	form
sep	'/'

objectgroupsNULL

<bates@stat.wisc.edu>

```
vf1 <- varPower(form = ~ age | Sex)
vf1 <- Initialize(vf1, Orthodont)
getGroups(vf1)
```

getGroupsFormula

`formula(object) | asList`

`getGroupsFormula(object, asList, sep)`

<code>object</code>	
<code>asList</code>	<code>TRUE FALSE FALSE</code>
<code>sep</code>	<code>'/'</code>

`formula(object) formula(object) NULL`

`<bates@stat.wisc.edu>`

[getGroupsFormula.gls](#) [getGroupsFormula.lmList](#) [getGroupsFormula.lme](#)
[getGroupsFormula.reStruct](#) [getGroups](#)

`getGroupsFormula(y ~ x | g1/g2)`

getResponse

`data.frame gls lme lmList`

`getResponse(object, form)`

<code>object</code>	
<code>form</code>	<code>formula(object)</code>

`data.frame form object`
`gls lme lmList fitted residuals`

`<bates@stat.wisc.edu>`

[getResponseFormula](#)

`getResponse(Orthodont)`

getResponseFormula

formula{object}

getResponseFormula(object)

object

formula{object}

<bates@stat.wisc.edu>

[getResponse](#)

getResponseFormula(y ~ x | g)

getVarCov

```
getVarCov(obj, ...)
## S3 method for class 'lme'
getVarCov(obj, individuals,
  type = c("random.effects", "conditional", "marginal"), ...)
## S3 method for class 'gls'
getVarCov(obj, individual = 1, ...)
```

obj	lme gls
individuals	lme
individual	gls individual
type	lme type"random.effects""conditional""marginal"
...	

<lindstro@biostat.wisc.edu>

lme_{gls}

```
fm1 <- lme(distance ~ age, data = Orthodont, subset = Sex == "Female")
getVarCov(fm1)
getVarCov(fm1, individuals = "F01", type = "marginal")
getVarCov(fm1, type = "conditional")
fm2 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
getVarCov(fm2)
```

gls

```
gls(model, data, correlation, weights, subset, method, na.action,
     control, verbose)
## S3 method for class 'gls'
update(object, model., ..., evaluate = TRUE)
```

object	"gls"
model	~+
model.	update.formula
data	modelcorrelationweightssubsetgls
correlation	corStruct corClasses corStructform corStructNULL
weights	varFunc varFixed varClasses varFuncNULL
subset	data
method	"REML" "ML" "REML"
na.action	NA.fail gls
control	glsControl
verbose	TRUEFALSE
...	
evaluate	TRUE

[offset](#)model

"gls"[print](#)[plot](#)[summary](#)[glsObject](#)[resid](#)[coeff](#)[fitted](#)

<bates@stat.wisc.edu>

correlation

[corClassesglsControlglsObjectglsStructplot.glspredict.glsqqnorm.gls](#)
[residuals.glssummary.glsvarClassesvarFunc](#)

```
# AR(1) errors within each Mare
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
# variance increases as a power of the absolute fitted values
fm2 <- update(fm1, weights = varPower())
```

glsControl

controlgls

```
glsControl(maxIter, msMaxIter, tolerance, msTol, msVerbose,
            singular.ok, returnObject = FALSE, apVar, .relStep,
            opt = c("nlsminb", "optim"), optimMethod,
            minAbsParApVar, natural, sigma = NULL)
```

maxIter	gls
msMaxIter	optgls
tolerance	gls
msTol	optim
msVerbose	traceoptFALSE
singular.ok	FALSE
returnObject	FALSE
apVar	TRUE
.relStep	.Machine\$double.eps^(1/3)
opt	"nlsminb""optim"
optimMethod	optim"BFGS""L-BFGS-B"
minAbsParApVar	0.05
natural	TRUE
sigma	NULL0

<bates@stat.wisc.edu>sigma

gls

```
# decrease the maximum number of iterations and request tracing
glsControl(msMaxIter = 20, msVerbose = TRUE)
```

glsObject

gls"glsvanovacoeffittedformulagetGroupsgetResponseintervalslogLikplotpredict
printresidualssummaryupdate

"gls"

apVar	apVar = FALSEglsNULL
call	gls
coefficients	
contrasts	
dims	Np
fitted	
modelStruct	glsStructcorStructvarFunc
groups	
logLik	
method	"ML""REML"
numIter	
residuals	
sigma	
varBeta	

<bates@stat.wisc.edu>

glsStruct

glsStruct

glsStruct corStruct varFunc NULL glsStruct

glsStruct(corStruct, varStruct)

corStruct	corStruct NULL
varStruct	varFunc NULL

<bates@stat.wisc.edu>

[corClasses](#) [glsresiduals](#) [glsStruct](#) [varFunc](#)

gls1 <- glsStruct(corAR1(), varPower())

Glucose

Glucose

623514

2am 6am 10am 2pm 6pm 10pm

Glucose2

Glucose2

17

12

gnls

```
gnls(model, data, params, start, correlation, weights, subset,  
      na.action, naPattern, control, verbose)
```

model	~data
data	modelcorrelationweightssubsetnaPatterngnls
params	p1+...+pn~x1+...+xmp1~x1+...+xmp1,...,pnmodelx1+...+xm1start
start	modelselfStartingmodelnls
correlation	corStruct corClasses corStructformcorStructNULL
weights	varFuncvarFixed varClasses varFuncNULL
subset	data
na.action	NAna.failgnls
naPattern	
control	gnlsControl
verbose	TRUEFALSE

```
gnlsglsprintplotsummarygnlsObjectresidcoeffitted
```

<bates@stat.wisc.edu>

correlation

[corClasses](#)[gnlsControl](#)[gnlsObject](#)[gnlsStruct](#)[predict.gnls](#)[varClasses](#)[varFunc](#)

```
# variance increases with a power of the absolute fitted values
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
summary(fm1)
```

[gnlsControl](#)

controlgnls

```
gnlsControl(maxIter = 50, nlsMaxIter = 7, msMaxIter = 50, minScale = 0.001,
            tolerance = 1e-6, nlsTol = 0.001, msTol = 1e-7,
            returnObject = FALSE, msVerbose = FALSE,
            apVar = TRUE, .relStep =,
            opt = c("nlsminb", "optim"), optimMethod = "BFGS",
            minAbsParApVar = 0.05, sigma = NULL)
```

maxIter	gnls
nlsMaxIter	nlsgnls
msMaxIter	optgnls
minScale	nls
tolerance	gnls
nlsTol	nls
msTol	optim
returnObject	warningstop()
msVerbose	traceoptFALSE


```
apVar          TRUE
.relStep       .Machine$double.eps^(1/3)
opt            "nlsminb" "optim"
optimMethod    "optim" "BFGS" "L-BFGS-B"
minAbsParApVar 0.05
sigma          NULL
```

```
<bates@stat.wisc.edu>sigma
```

[gnls](#)

```
# decrease the maximum number of iterations and request tracing
gnlsControl(msMaxIter = 20, msVerbose = TRUE)
```

gnlsObject

```
gnls"gnls""gls"anovacoefffittedformulagetGroupsgetResponseintervalslogLikplot
predictprintresidualssummaryupdate
```

```
"gnls"
```

```
apVar          apVar = FALSEgnlsNULL
call           gnls
coefficients
contrasts
dims           Np
fitted
modelStruct    gnlsStructcorStructvarFunc
groups
logLik
numIter
plist
pmap
residuals
sigma
varBeta
```

```
<bates@stat.wisc.edu>
```

[gnls](#)gnlsStruct

gnlsStruct

gnlsStructcorStructvarFuncNULLgnlsStruct

gnlsStruct(corStruct, varStruct)

corStruct	corStructNULL
varStruct	varFuncNULL

<bates@stat.wisc.edu>

[gnlscorClassesresiduals.gnlsStructvarFunc](#)

gnls1 <- gnlsStruct(corAR1(), varPower())

groupedData

groupedDataformulaformulaouterinnerlabelsunitsorder.groupsTRUEFUN
nfnGroupedDataanffGroupedDataanmGroupedDatagroupedData

groupedData(formula, data, order.groups, FUN, outer, inner,
 labels, units)

```
## S3 method for class 'groupedData'  
update(object, formula, data, order.groups, FUN,  
       outer, inner, labels, units, ...)
```

object	groupedData
formula	resp ~ cov grouprespcovgroup1/fact1/fact2fact2fact1
data	formulagroupedData
order.groups	FUNTRUE
FUN	order.groups = TRUEmax
outer	groupedDataouter = TRUEouterNULL
inner	groupedDataNULL
labels	xy
units	xy
...	

nfnGroupedData
nffGroupedData
nmGroupedData
groupedData
data.frame

[formula](#)
[apply](#)
[gsummary](#)
[lme](#)
[plot.nffGroupedData](#)
[plot.nfnGroupedData](#)
[plot.nmGroupedData](#)
[areStruct](#)

```
Orth.new <- # create a new copy of the groupedData object
  groupedData( distance ~ age | Subject,
    data = as.data.frame( Orthodont ),
    FUN = mean,
    outer = ~ Sex,
    labels = list( x = "Age",
      y = "Distance from pituitary to pterygomaxillary fissure" ),
    units = list( x = "(yr)", y = "(mm)" ) )
plot( Orth.new )      # trellis plot by Subject
formula( Orth.new )   # extractor for the formula
gsummary( Orth.new )  # apply summary by Subject
fm1 <- lme( Orth.new ) # fixed and groups formulae extracted from object
Orthodont2 <- update(Orthodont, FUN = mean)
```

`gsummary`

`groupedData`

`gsummary(object, FUN, omitGroupingFactor, form, level, groups, invariantsOnly, ...)`

<code>object</code>	<code>groupedData</code> <code>data.frame</code>
<code>FUN</code>	<code>object</code> <code>groups</code> <code>FUN</code> <code>Orderedfactor</code> <code>numeric</code> <code>mean</code> <code>Mode</code> <code>factor</code> <code>orderedMode</code> <code>gsummary</code> <code>mode</code>
<code>omitGroupingFactor</code>	<code>TRUE</code> <code>FALSE</code>
<code>form</code>	<code>object</code> <code>formula(object)</code>
<code>level</code>	
<code>groups</code>	<code>getGroups(object, form, level)</code>
<code>invariantsOnly</code>	<code>TRUE</code> <code>object</code> <code>omitGroupingFactor = TRUE</code> <code>FALSE</code>
<code>...</code>	<code>na.rm = TRUE</code>

data.frameobject

<bates@stat.wisc.edu>

[summarygroupedDatagetGroups](#)

```
gsummary(Orthodont) # default summary by Subject
## gsummary with invariantsOnly = TRUE and omitGroupingFactor = TRUE
## determines whether there are covariates like Sex that are invariant
## within the repeated observations on the same Subject.
gsummary(Orthodont, invariantsOnly = TRUE, omitGroupingFactor = TRUE)
```

Gun

Gun

M1M2

T1ST3ST2ST1AT2AT3AT1HT3HT2H

SlightAverageHeavy

IGF

IGF

Initialize

corStructglsStructlmeStructreStructvarFunc

Initialize(object, data, ...)

object corStructvarFunc
data
...

object

<bates@stat.wisc.edu>

[Initialize.corStructInitialize.glsStructInitialize.lmeStructInitialize.reStructInitialize.varFuncisInitialized](#)

Initialize.corStruct

```
objectdatacorStructDimobject

## S3 method for class 'corStruct'
Initialize(object, data, ...)

object      "corStruct"
data        formula(object)
...

object

<bates@stat.wisc.edu>
```

Dim.corStruct

```
cs1 <- corAR1(form = ~ 1 | Subject)
cs1 <- Initialize(cs1, data = Orthodont)
```

Initialize.glsStruct

```
glStruct

## S3 method for class 'glStruct'
Initialize(object, data, control, ...)

object      "glStruct"corStructvarFunc
data        formula(object)
control      glslist(singular.ok = FALSE)
...

glStructobject

<bates@stat.wisc.edu>
```

glStructInitialize.corStructInitialize.varFuncInitialize

Initialize.lmeStruct

lmeStruct

```
## S3 method for class 'lmeStruct'  
Initialize(object, data, groups, conLin, control, ...)
```

```
object      "lmeStruct"reStructcorStructvarFunc  
data        formula(object)  
groups      object  
conLin      "Xy"Xy"logLik"attr(object, "conLin")  
control     lmeList(niterEM=20, gradHess=TRUE)reStructobject  
...
```

lmeStructobject

<bates@stat.wisc.edu>

[lmeInitialize.reStructInitialize.corStructInitialize.varFuncInitialize](#)

Initialize.reStruct

pdMatobjectcontrol

```
## S3 method for class 'reStruct'  
Initialize(object, data, conLin, control, ...)
```

```
object      "reStruct"pdMat  
data        formula(object)  
conLin      "Xy"Xy"logLik"  
control     niterEMInitializelist(niterEM = 20)  
...
```

reStructobjectpdMat

<bates@stat.wisc.edu>

[reStructpdMatInitialize](#)

Initialize.varFunc

```
objectdataobjectobjectobjectcoef(object)
```

```
## S3 method for class 'varFunc'  
Initialize(object, data, ...)
```

```
object      "varFunc"  
data        formula(object)  
...
```

```
object
```

```
<bates@stat.wisc.edu>
```

Initialize

```
vf1 <- varPower( form = ~ age | Sex )  
vf1 <- Initialize( vf1, Orthodont )
```

intervals

```
objectglslmelmList
```

```
intervals(object, level, ...)
```

```
object  
level  
...
```

```
<bates@stat.wisc.edu>
```

[intervals.lme](#)[intervals.lmList](#)[intervals.gls](#)

intervals.gls

object

```
## S3 method for class 'glS'  
intervals(object, level, which, ...)
```

```
object      "glS"  
level  
which       "all""var-cov""coef""all"  
...
```

lowerest.upper

```
coef        which"var-cov"  
corStruct   which"coef"object  
varFunc     which"coef"object  
sigma
```

<bates@stat.wisc.edu>

[glSintervalsprint.intervals.gls](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,  
           correlation = corAR1(form = ~ 1 | Mare))  
intervals(fm1)
```

intervals.lme

objectpdNatural

```
## S3 method for class 'lme'
intervals(object, level = 0.95,
          which = c("all", "var-cov", "fixed"), ...)
```

```
object      "lme"
level
which       "all" "var-cov" "fixed" "all"
...
```

lowerest.upper

```
fixed       which"var-cov"
reStruct    which"fixed"
corStruct   which"fixed"object
varFunc     which"fixed"object
sigma
```

<bates@stat.wisc.edu>

[lmeintervalsprint.intervals.lmepdNatural](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
intervals(fm1)
```

`intervals.lmList`

```
lmobjectobjectlowerest.upper
```

```
## S3 method for class 'lmList'  
intervals(object, level = 0.95, pool = attr(object, "pool"), ...)
```

```
object      "lmList"lm  
level  
pool        attr(object, "pool")  
...
```

```
lmobject
```

```
<bates@stat.wisc.edu>
```

```
lmListintervalsplot.intervals.lmList
```

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)  
intervals(fm1)
```

`isBalanced`

```
isBalanced(object, countOnly, level)
```

```
object      groupedData  
countOnly   FALSE  
level
```

countOnlyFALSE

TRUEFALSE

<bates@stat.wisc.edu>

tablegroupedData

```
isBalanced(Orthodont)           # should return TRUE
isBalanced(Orthodont, countOnly = TRUE) # should return TRUE
isBalanced(Pixel)               # should return FALSE
isBalanced(Pixel, level = 1)    # should return FALSE
```

isInitialized

objectInitialize

isInitialized(object)

object

object

Initialize

```
pd1 <- pdDiag(~age)
isInitialized(pd1)
```

LDEsysMat

```
LDEsysMat(pars, incidence)
```

```
pars  
incidence      FromToParParlength(pars)FromTo
```

```
kk
```

```
kk
```

```
<bates@stat.wisc.edu>
```

```
# incidence matrix for a two compartment open system  
incidence <-  
  matrix(c(1,1,2,2,2,1,3,2,0), ncol = 3, byrow = TRUE,  
    dimnames = list(NULL, c("Par", "From", "To")))  
incidence  
LDEsysMat(c(1.2, 0.3, 0.4), incidence)
```

lme

```
lme.lmListlme.groupedData
```

```
lme(fixed, data, random, correlation, weights, subset, method,  
    na.action, control, contrasts = NULL, keep.data = TRUE)
```

```
## S3 method for class 'formula'  
lme(fixed, data, random, correlation, weights, subset, method,  
    na.action, control, contrasts = NULL, keep.data = TRUE)
```

```
## S3 method for class 'lme'  
update(object, fixed., ..., evaluate = TRUE)
```

object	lme
fixed	~+"lmList""groupedData" resp ~ 1resp ~ 03.1-112
fixed.	update.formula
data	fixedrandomcorrelationweightssubsetlme
random	~ x1 + ... + xn g1/.../gmx1 + ... + xng1/.../gmm/~ x1 + ... + xn g~ x1 + ... + xnpdMatNULLformula(object)pdMat"groupedData" pdMatreStructpdClassespdMatfixed
correlation	corStruct corClasses corStructNULL
weights	varFunc varFixed varClasses varFunc NNULL
subset	data
method	"REML""ML""REML"
na.action	NA na.fail lme
control	lmeControl
contrasts	contrasts.arg model.matrix.default
keep.data	data
...	
evaluate	TRUE

[offset](#)fixed

"lme"printplotsummary[lmeObject](#)residcoeffittedfixed.effectsrandom.effects

<bates@stat.wisc.edu>

correlation

[corClasses](#)
[lme.lmList](#)
[lme.groupedData](#)
[lmeControl](#)
[lmeObject](#)
[lmeStruct](#)
[lmList](#)
[pdClasses](#)
[plot.lmepredict](#)
[lmeqqnorm](#)
[lmeresiduals](#)
[lmeStructs](#)
[simulate.lmesummary.lme](#)
[varClasses](#)
[varFunc](#)

```

fm1 <- lme(distance ~ age, data = Orthodont) # random is ~ age
fm2 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
summary(fm1)
summary(fm2)

```

[lme.groupedData](#)

[formula\(fixed\)](#)
[groupedData](#)
[fixeddata](#)
[lme.formula](#)
[lme.formula](#)

```

## S3 method for class 'groupedData'
lme(fixed, data, random, correlation, weights,
    subset, method, na.action, control, contrasts, keep.data = TRUE)

```

fixed	"groupedData"
data	
random	$\sim x_1 + \dots + x_n \mid g_1 / \dots / g_m x_1 + \dots + x_n g_1 / \dots / g_m m / \sim x_1 + \dots + x_n \mid g$ $\sim x_1 + \dots + x_n$ <p>pdMatNULL formula(object) pdMat groupedData pdMat reStruct pdClasses pdMat fixed</p>
correlation	corStruct corClasses corStruct NULL
weights	varFunc varFixed varClasses varFunc NULL
subset	data
method	"REML" "ML" "REML"
na.action	NA na.fail lme
control	lmeControl
contrasts	contrasts.arg model.matrix.default
keep.data	data

[lmeprint](#)
[plot](#)
[summary](#)
[lmeObject](#)
[resid](#)
[coeff](#)
[fitted](#)
[fixed.effects](#)
[random.effects](#)

<bates@stat.wisc.edu>

correlation

[lmeGroupedData](#)[lmeObject](#)

```
fm1 <- lme(Orthodont)
summary(fm1)
```

[lme.lmList](#)

[randomlmList](#)[randomformula\(fixed\)](#)[datafixedfixeddata](#)[lme.formula](#)[lme.formula](#)

```
## S3 method for class 'lmList'
lme(fixed, data, random, correlation, weights, subset, method,
    na.action, control, contrasts, keep.data)
```

fixed	"lmList." lm
data	
random	pdMatformulaformula(fixed)
correlation	corStruct corClasses corStructNULL
weights	varFuncvarFixed varClasses varFuncNULL
subset	data
method	"REML""ML""REML"
na.action	NAAna.faillme
control	lmeControl
contrasts	contrasts.argmodel.matrix.default
keep.data	data

[lmeprint](#)[plotsummary](#)[lmeObject](#)[residcoeffittedfixed.effects](#)[random.effects](#)

<bates@stat.wisc.edu>

correlation

[lme1mListlmeObject](#)

```
fm1 <- lmeList(Orthodont)
fm2 <- lme(fm1)
summary(fm1)
summary(fm2)
```

lmeControl

lmeControl()[listcontrol](#)lme

```
lmeControl(maxIter = 50, msMaxIter = 50, tolerance = 1e-6, niterEM = 25,
            msMaxEval = 200,
            msTol = 1e-7, msVerbose = FALSE,
            returnObject = FALSE, gradHess = TRUE, apVar = TRUE,
            .relStep = .Machine$double.eps^(1/3), minAbsParApVar = 0.05,
            opt = c("nlminb", "optim"),
            optimMethod = "BFGS", natural = TRUE,
            sigma = NULL,
            allow.n.lt.q = FALSE,
            ...)
```

maxIter	lme50
msMaxIter	lme50
tolerance	lme1e-6
niterEM	25
msMaxEval	nlminb 200

```

msTol          optim1e-7
msVerbose      tracenlminboptimFALSE
returnObject   warningstop()FALSE
gradHess       corStructvarFuncpdMatpdSymmpdDiagpdIdentpdCompSymmTRUE
apVar          TRUE
.relStep       .Machine$double.eps^(1/3)
opt            "nlminb""optim"
optimMethod    optim"BFGS""L-BFGS-B"
minAbsParApVar 0.05
natural        pdNaturalpdSymmreStructTRUE
sigma          NULL0
allow.n.lt.q   logicalFALSENA
...            optnlminbabs.toloptimtracemaxitreltol

```

```
<bates@stat.wisc.edu>sigma
```

```
lmenlminboptim
```

```

# decrease the maximum number iterations in the ms call and
# request that information on the evolution of the ms iterations be printed
str(lCtr <- lmeControl(msMaxIter = 20, msVerbose = TRUE))
## This should always work:
do.call(lmeControl, lCtr)

```

```
lmeObject
```

```

lme"lme"anovacoeffittedfixed.effectsformulagetGroupsgetResponseintervalslogLik
pairsplotpredictprinrandom.effectsresidualssigmasummaryupdatevcov

```

```
"lme"
```

```

apVar          apVar = FALSElmeNULL
call          lme
coefficients   fixedrandomrandom
contrasts      contrasts
dims           NQqvecngrpsncol
fitted         fitted

```

```
fixDF          Xterms
groups
logLik
method         "ML""REML"
modelStruct    lmeStructreStructcorStructvarFunc
numIter
residuals
terms          termsformula
sigma
varFix
```

<bates@stat.wisc.edu>

[lme](#)lmeStruct

lmeStruct

lmeStructreStructcorStructvarFuncNULLlmeStruct

lmeStruct(reStruct, corStruct, varStruct)

```
reStruct       reStruct
corStruct      corStructNULL
varStruct      varFuncNULL
```

<bates@stat.wisc.edu>

[corClasseslmeresiduals.lmeStructreStructvarFunc](#)

```
lms1 <- lmeStruct(reStruct(~age), corAR1(), varPower())
```

lmList

Dataglmdataobject

```
lmList(object, data, level, subset, na.action = na.fail,  
        pool = TRUE, warn.lm = TRUE)
```

```
## S3 method for class 'formula'  
lmList(object, data, level, subset, na.action = na.fail,  
        pool = TRUE, warn.lm = TRUE)
```

```
## S3 method for class 'lmList'  
update(object, formula., ..., evaluate = TRUE)  
## S3 method for class 'lmList'  
print(x, pool, ...)
```

object	lmListy ~ x1+...+xn ggroupedDatayx1,...,xnglmgdatagroupedData lmList.groupedData update.lmListobjectlmList
formula	update.lmListlm
formula.	update.formula
data	object
level	
subset	data
na.action	NAna.faillmList
pool	
warn.lm	logical lm()tryCatchwarning
x	lmList
...	
evaluate	TRUE

lmcoeffixed.effectslmepairsplotpredictrandom.effectssummaryupdatelmList

[lm](#)me.lmListplot.lmListpooledSDpredict.lmListresiduals.lmListsummary.lmList

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)  
summary(fm1)
```

lmList.groupedData

formula(object)groupedDataobjectdata`lmList.formula``lmList.formula`

```
## S3 method for class 'groupedData'
lmList(object, data, level, subset, na.action = na.fail,
        pool = TRUE, warn.lm = TRUE)
```

```
object      "groupedData"
data
level
subset      data
na.action    NA na.fail
poolwarn.lm  logicallmList
```

`lmcoeffixed.effects``lmepairs``plot``predict``random.effects``summary``update``lmList`

`groupedData``lme``lmList``lmList``lmList``formula`

```
fm1 <- lmList(Orthodont)
summary(fm1)
```

logDet

`corStruct``pdMat``reStruct`

`logDet(object, ...)`

```
object
...
```

<bates@stat.wisc.edu>

`logLik``logDet``corStruct``logDet``pdMat``logDet``reStruct`

logDet.corStruct

objectobject

```
## S3 method for class 'corStruct'  
logDet(object, covariate, ...)
```

```
object          "corStruct"  
covariate       getCovariate(object)  
...
```

objectobject

<bates@stat.wisc.edu>

[logLik.corStructcorMatrix.corStructlogDet](#)

```
cs1 <- corAR1(0.3)  
logDet(cs1, covariate = 1:4)
```

logDet.pdMat

object

```
## S3 method for class 'pdMat'  
logDet(object, ...)
```

```
object          "pdMat"  
...
```

object

<bates@stat.wisc.edu>

[pdMatlogDet](#)

```
pd1 <- pdSymm(diag(1:3))  
logDet(pd1)
```

logDet.reStruct

pdMatobject

```
## S3 method for class 'reStruct'  
logDet(object, ...)
```

```
object          "reStruct"pdMat  
...
```

pdMatobject

[reStructpdMatlogDet](#)

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),  
  B = pdDiag(2 * diag(4), form = ~Educ)))  
logDet(rs1)
```

logLik.corStruct

object

```
## S3 method for class 'corStruct'  
logLik(object, data, ...)
```

```
object          "corStruct"  
data            logLik  
...
```

object

<bates@stat.wisc.edu>

[logDet.corStructlogLik.lme](#)

```
cs1 <- corAR1(0.2)  
cs1 <- Initialize(cs1, data = Orthodont)  
logLik(cs1)
```

logLik.glsStruct

```
Parsobjectsettingsobject

## S3 method for class 'glStruct'
logLik(object, Pars, conLin, ...)

object          "glStruct"corStruct"varFunc"
Pars
conLin          "Xy"Xy"logLik"attr(object, "conLin")
...

objectPars

<bates@stat.wisc.edu>

glsglsStructlogLik.lme
```

logLik.gnls

```
object

## S3 method for class 'gnls'
logLik(object, REML, ...)

object          "gnls"
REML            logLik,glsFALSE
...

object

<bates@stat.wisc.edu>

gnlslogLik.lme

fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
logLik(fm1)
```

logLik.gnlsStruct

Parsobject

```
## S3 method for class 'gnlsStruct'  
logLik(object, Pars, conLin, ...)
```

```
object          gnlsStructcorStructvarFunc  
Pars  
conLin          "Xy"Xy"logLik"attr(object, "conLin")  
...
```

objectPars

<bates@stat.wisc.edu>

[gnlsgnlsStructlogLik.gnls](#)

logLik.lme

REML=FALSEobject

```
## S3 method for class 'lme'  
logLik(object, REML, ...)
```

```
object          "lme"  
REML            TRUEFALSETRUEobjectmethod = "REML"  
...
```

object

```
lmeGlslogLik.corStructlogLik.glsStructlogLik.lmeStructlogLik.lmList
logLik.reStructlogLik.varFunc
```

```
fm1 <- lme(distance ~ Sex * age, Orthodont, random = ~ age, method = "ML")
logLik(fm1)
logLik(fm1, REML = TRUE)
```

```
logLik.lmeStruct
```

```
Parsobjectsettingsobject
```

```
## S3 method for class 'lmeStruct'
logLik(object, Pars, conLin, ...)
```

```
object          "lmeStruct"reStructcorStructvarFunc
Pars
conLin           "Xy"Xy"logLik"attr(object, "conLin")
...
```

```
objectPars
```

```
<bates@stat.wisc.edu>
```

```
lmeLmeStructlogLik.lme
```

logLik.lmList

```
pool=FALSElmobjectlmobject

## S3 method for class 'lmList'
logLik(object, REML, pool, ...)

object          "lmList"lm
REML            TRUEFALSEFALSE
pool            lmobjectattr(object, "pool")
...

lmobjectlmobject

<bates@stat.wisc.edu>

lmListlogLik.lme

fm1 <- lmList(distance ~ age | Subject, Orthodont)
logLik(fm1)  # returns NA when it should not
```

logLik.reStruct

```
objectconLincoef(object)settingsobject

## S3 method for class 'reStruct'
logLik(object, conLin, ...)

object          "reStruct"pdMat
conLin          "Xy"Xy"logLik"
...

objectconLincoef{object}

<bates@stat.wisc.edu>

reStructpdMatlogLik.lme
```

logLik.varFunc

object

```
## S3 method for class 'varFunc'  
logLik(object, data, ...)
```

```
object      "varFunc"  
data        logLik  
...
```

object

<bates@stat.wisc.edu>

[logLik.lme](#)

```
vf1 <- varPower(form = ~age)  
vf1 <- Initialize(vf1, Orthodont)  
coef(vf1) <- 0.1  
logLik(vf1)
```

Machines

Machines

ABC

MathAchieve

MathAchieve

NoYes

MaleFemale

summary(MathAchieve)

MathAchSchool

MathAchSchool

PublicCatholic

01

MathAchieve

Matrix

pdMatpdBlockedreStruct

matrix(object) <- value

object as.matrix

value as.matrix(object)object

<bates@stat.wisc.edu>

as.matrixmatrix<-.pdMatmatrix<-.reStruct

Matrix.pdMat

objectvaluevalueNULL

```
## S3 replacement method for class 'pdMat'
matrix(object) <- value
## S3 replacement method for class 'pdBlocked'
matrix(object) <- value
```

object "pdMat"

value objectas.matrix(object)

pdMatpdBlockedobjectvalue

<bates@stat.wisc.edu>

pdMat"matrix<-"

```
class(pd1 <- pdSymm(diag(3))) # "pdSymm" "pdMat"
matrix(pd1) <- diag(1:3)
pd1
```

Matrix.reStruct

```
valuepdMatobject

## S3 replacement method for class 'reStruct'
matrix(object) <- value

object      "reStruct"pdMat
value      pdMatobject

reStructobjectpdMatvalue

<bates@stat.wisc.edu>

reStructpdMat"matrix<-"

rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
matrix(rs1) <- list(diag(2), 3)
```

Meat

Meat

IIVIIIIIV
II-1IV-1

Milk

Milk

barleybarley+lupinslupins

`model.matrix.reStruct`

```
formula(object)datalength(object) > 1
```

```
## S3 method for class 'reStruct'  
model.matrix(object, data, contrast, ...)
```

```
object      "reStruct"pdMat  
data        formula(object)  
contrast    factordatadatacontrasts  
...
```

```
formula(object)
```

```
<bates@stat.wisc.edu>
```

```
model.matrixcontrastsreStructformula.reStruct
```

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)  
model.matrix(rs1, Pixel)
```

Muscle

Muscle

2

Names

formulapdBlockedpdMatreStruct

Names(object, ...)
Names(object, ...) <- value

object
...
value object

objectvalue

<bates@stat.wisc.edu>

[Names.formulaNames.pdBlockedNames.pdMatNames.reStruct](#)

Names.formula

objectmodel.matrix

```
## S3 method for class 'formula'  
Names(object, data, exclude, ...)
```

```
object      "formula"  
data        objectNames.formula  
exclude     c("pi", ".")  
...
```

model.matrixobjectexcluded

<bates@stat.wisc.edu>

[model.matrixtermsNames](#)

```
Names(distance ~ Sex * age, data = Orthodont)
```

Names.pdBlocked

Dimnamesobject

```
## S3 method for class 'pdBlocked'  
Names(object, asList, ...)
```

```
object      "pdBlocked"  
asList      TRUElistFALSEFALSE  
...
```

asListFALSEobjectasListTRUEobject

<bates@stat.wisc.edu>

[NamesNames.pdMat](#)

```
pd1 <- pdBlocked(list(~Sex - 1, ~age - 1), data = Orthodont)  
Names(pd1)
```

Names.pdMat

Dimnamesobjectobject

```
## S3 method for class 'pdMat'
Names(object, ...)
## S3 replacement method for class 'pdMat'
Names(object, ...) <- value
```

```
object      "pdMat"
value       objectobjectobject
...
```

objectDimnamesNULL

Dimnamesobjectlist(value, value)

<bates@stat.wisc.edu>

[NamesNames.pdBlocked](#)

```
pd1 <- pdSymm(~age, data = Orthodont)
Names(pd1)
```

Names.reStruct

pdMatobject

```
## S3 method for class 'reStruct'
Names(object, ...)
## S3 replacement method for class 'reStruct'
Names(object, ...) <- value
```

```
object      "reStruct"pdMat
value       pdMatobjectobject
...
```

pdMatobject

NamespdMatobjectvalue

<bates@stat.wisc.edu>

[reStructpdMatNames.pdMat](#)

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
Names(rs1)
```

needUpdate

needUpdateobjectNULLFALSEFALSETRUE

needUpdate(object)

object

object

<bates@stat.wisc.edu>

[needUpdate.modelStruct](#)

```
vf1 <- varExp()
vf1 <- Initialize(vf1, data = Orthodont)
needUpdate(vf1)
```

`needUpdate.modelStruct`

`object`

```
## S3 method for class 'modelStruct'  
needUpdate(object)
```

```
object          "modelStruct"corStructvarFunc
```

`object`

`<bates@stat.wisc.edu>`

[needUpdate](#)

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),  
  varStruct = varPower(form = ~age))  
needUpdate(lms1)
```

Nitrendipene

Nitrendipene

2134

nlme

```
nlme(model, data, fixed, random, groups, start, correlation, weights,
      subset, method, na.action, naPattern, control, verbose)
```

```
## S3 method for class 'formula'
```

```
nlme(model, data, fixed, random, groups, start, correlation, weights,
      subset, method, na.action, naPattern, control, verbose)
```

model	~nlsListdata <code>nlme.nlsList</code>
data	modelfixedrandomcorrelationweightssubsetnaPatternnlme
fixed	f1+...+fn~x1+...+xm f1~x1+...+xm f1,...,fn model x1+...+xm1
random	r1+...+rn~x1+...+xm g1/.../gQ r1,...,rn model x1+...+xm g1/.../gQ Q/r1+...+rn~x1+...+xm r1~x1+...+xm pdMat NULL formula(random) groups groupedData pdMat reStruct <code>pdClasses</code> pdMat fixed
groups	~g1~g1/.../gQ g1,...,gQ data g1g2
start	fixed fixed selfStart nlsList random
correlation	corStruct <code>corClasses</code> corStruct NULL
weights	varFunc varFixed <code>varClasses</code> varFunc NULL
subset	data
method	"REML" "ML" "ML"
na.action	NA na.fail nlme
naPattern	
control	nlmeControl
verbose	TRUE FALSE

```
nlme print plot summary nlmeObject resid coeff fitted fixed.effects random.effects
```

<bates@stat.wisc.edu>

[lme](#)

[nlmeControl](#)[nlme.nlsList](#)[nlmeObject](#)[nlslList](#)[nlmeStruct](#)[pdClasses](#)[reStruct](#)[varFunc](#)
[corClasses](#)[varClasses](#)

```
fm1 <- nlme(height ~ SSasyp(age, Asym, R0, lrc),
            data = Loblolly,
            fixed = Asym + R0 + lrc ~ 1,
            random = Asym ~ 1,
            start = c(Asym = 103, R0 = -8.5, lrc = -3.3))
summary(fm1)
fm2 <- update(fm1, random = pdDiag(Asym + lrc ~ 1))
summary(fm2)
```

[nlme.nlsList](#)

[random](#)[lml](#)[istr](#)[random](#)[formula\(fixed\)](#)[data](#)[fixed](#)[fixed](#)[data](#)[nlme.formula](#)[nlme.formula](#)

```
## S3 method for class 'nlsList'
nlme(model, data, fixed, random, groups, start, correlation, weights,
      subset, method, na.action, naPattern, control, verbose)
```

model	" nlsList "nls
data	
fixed	
random	pdMatformulaformula(fixed)
groups	~g1~g1/.../gQg1,...,gQdatag1g2
start	fixedfixedselfStartnlsListrandom
correlation	corStruct corClasses corStructNULL
weights	varFuncvarFixed varClasses varFuncNULL
subset	data
method	"REML""ML""ML"
na.action	NAAna.failnlme
naPattern	
control	nlmeControl
verbose	TRUEFALSE

[nlmeprint](#)[plot](#)[summary](#)[nlmeObject](#)[resid](#)[coeff](#)[fitted](#)[fixed.effects](#)[random.effects](#)

<bates@stat.wisc.edu>

correlation

[nlme1mListnlmeObject](#)

```
fm1 <- nlsList(SSasyp, data = Loblolly)
fm2 <- nlme(fm1, random = Asym ~ 1)
summary(fm1)
summary(fm2)
```

nlmeControl

controlnlme

```
nlmeControl(maxIter, pnlsMaxIter, msMaxIter, minScale,
            tolerance, niterEM, pnlsTol, msTol,
            returnObject, msVerbose, msWarnNoConv,
            gradHess, apVar, .relStep, minAbsParApVar = 0.05,
            opt = c("nlminb", "nlm"), natural = TRUE, sigma = NULL, ...)
```

maxIter	nlme
pnlsMaxIter	PNLSnlme
msMaxIter	nlminb iter.maxnlmiterlimnlme
minScale	PNLS0.001
tolerance	nlme1e-6
niterEM	
pnlsTol	PNLS1e-3
msTol	nlmgradtolnlm1e-7
returnObject	FALSE
msVerbose	tracenlminb(..., control= list(trace = *, ...))print.levelnlm() FALSE
msWarnNoConv	warning optTRUE


```

gradHess      nlmcStructvarFuncpdMatpdSymmpdDiagpdIdentpdCompSymmTRUE
apVar         TRUE
.relStep      .Machine$double.eps^(1/3)
minAbsParApVar 0.05
opt           "nlminb""nlm"
natural       pdNaturalpdSymmreStructTRUE
sigma         NULL
...           nlminbtraceiter.maxeval.maxabs.tol

```

<bates@stat.wisc.edu>sigma

nlmenlmoptimnlmeStruct

```

# decrease the maximum number of iterations and request tracing
nlmeControl(msMaxIter = 20, msVerbose = TRUE)

```

nlmeObject

```

nlme"nlme""lme"anovacoeffittedfixed.effectsformulagetGroupsgetResponseintervals
logLikpairsplotpredictprintrandom.effectsresidualssummaryupdate

```

"nlme"

```

apVar         apVar = FALSEnlmeNULL
call          nlme
coefficients   fixedrandomrandom
contrasts      contrasts
dims          NQqvecngrpsncol
fitted         fitted
fixDF          Xterms
groups         groups
logLik         logLik
map            fmaprmmaprmapRelbmap
method         "ML""REML"
modelStruct    nlmeStructreStructcorStructvarFunc
numIter        numIter
residuals      residuals
sigma          sigma
varFix         varFix

```

<bates@stat.wisc.edu>

[nlme](#)`nlmeStruct`

`nlmeStruct`

`nlmeStruct``reStruct``corStruct``varFunc``NULL``nlmeStruct`

`nlmeStruct(reStruct, corStruct, varStruct)`

<code>reStruct</code>	<code>reStruct</code>
<code>corStruct</code>	<code>corStruct</code> <code>NULL</code>
<code>varStruct</code>	<code>varFunc</code> <code>NULL</code>

<bates@stat.wisc.edu>

[corClasses](#)`nlmeresiduals``nlmeStruct``reStruct``varFunc`

`nls1 <- nlmeStruct(reStruct(~age), corAR1(), varPower())`

`nlsList`

`Data``model``nls``data``model`

`nlsList(model, data, start, control, level, subset,`
`na.action = na.fail, pool = TRUE, warn.nls = NA)`

`## S3 method for class 'formula'`
`nlsList(model, data, start, control, level, subset,`
`na.action = na.fail, pool = TRUE, warn.nls = NA)`

`## S3 method for class 'nlsList'`
`update(object, model., ..., evaluate = TRUE)`

```

object      nlsListnls
model       ~|selfStartnlsList.selfStart
model.      update.formula
data        model
start       modelstartnlsmodelselfStart
control     controlnls
level
subset      data
na.action   NAAna.failnlsList
pool        pool
warn.nls    logicalnls()tryCatchwarning
...
evaluate    TRUE

```

```
nls(.)
```

```

3.1-127tryCatchwarning"nlsList"warn.nls      =      FALSEwarn.nls      =      NA
getOption("show.error.messages")
nlsList()try(*)silent=FALSE)show.error.messages

```

```
nlscoeffixed.effects1mepairsplotpredictrandom.effectssummaryupdatenlsList
```

```
nlsnlme.nlsListnlsList.selfStartsummary.nlsList
```

```

fm1 <- nlsList(uptake ~ SSasymOff(conc, Asym, lrc, c0),
  data = C02, start = c(Asym = 30, lrc = -4.5, c0 = 52))
summary(fm1)
cfm1 <- confint(fm1) # via profiling each % FIXME: only *one* message instead of one *each*
mat.class <- class(matrix(1)) # ("matrix", "array") for R >= 4.0.0; ("matrix" in older R)
i.ok <- which(vapply(cfm1,
  function(r) identical(class(r), mat.class), NA))
stopifnot(length(i.ok) > 0, !anyNA(match(c(2:4, 6:9, 12), i.ok)))
## where as (some of) the others gave errors during profile re-fitting :
str(cfm1[- i.ok])

```

nlsList.selfStart

formula(data)modelnls

```
## S3 method for class 'selfStart'
nlsList(model, data, start, control, level, subset,
        na.action = na.fail, pool = TRUE, warn.nls = NA)
```

```
model      "selfStart" data
data       modelmodeldata"groupedData"
start      modelstartnlsmodelselfStart
control    controlnls
level
subset     data
na.action  NAna.failnlsList
poolwarn.nls logicalnlsList
```

```
nlsNULLnlscoeffixed.effects1mepairsplotpredictrandom.effectssummaryupdate
nlsList
```

[selfStartgroupedData](#)[nlsnlsListnlme.nlsListnlsList.formula](#)

```
fm1 <- nlsList(SSasymptOff, CO2)
summary(fm1)
```

Oats

Oats

VIVIIIIVIII
Golden RainMarvellousVictory

3×4

Orthodont

Orthodont

M01M16F01F13
MaleFemale

`formula(Orthodont)`
`plot(Orthodont)`

Ovary

Ovary

Oxboys

Oxboys

age

Oxide

Oxide

12

123

`pairs.compareFits`

`xtrellisxyplottrellissplom`

```
## S3 method for class 'compareFits'
pairs(x, subset, key, ...)
```

x	compareFits
subset	x
key	TRUEFALSEkeytrellissplomxyplotTRUE
...	trellis

[compareFitsplot.compareFitspairs.lmepairs.lmListxyplotsplom](#)

```
example(compareFits) # cF12 <- compareFits(coef(lmList(Orthodont)), .. lme(*))
pairs(cF12)
```

`pairs.lme`

`form||xyplotsplom`

```
## S3 method for class 'lme'
pairs(x, form, label, id, idLabels, grid, ...)
```

x	"lme"
form	xx". " form ~ coef(.)
label	
id	1 - <i>value</i> idLabels
idLabels	id
grid	FALSE
...	

<bates@stat.wisc.edu>

[lmepairs.compareFitspairs.lmListxyplotsplom](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)

# scatter plot of coefficients by gender, identifying unusual subjects
pairs(fm1, ~coef(., augFrame = TRUE) | Sex, id = 0.1, adj = -0.5)

# scatter plot of estimated random effects :
pairs(fm1, ~ranef(.))
```

pairs.lmList

xform||xyplotsplom

```
## S3 method for class 'lmList'
pairs(x, form, label, id, idLabels, grid, ...)
```

x	"lmList"lm
form	xx"." form ~ coef(.) x
label	
id	1 - valueidLabels
idLabels	id
grid	FALSE
...	

<bates@stat.wisc.edu>

[lmListpairs.lmepairs.compareFitsxyplotsplom](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)

# scatter plot of coefficients by gender, identifying unusual subjects
pairs(fm1, ~coef(.) | Sex, id = 0.1, adj = -0.5)

# scatter plot of estimated random effects -- "bivariate Gaussian (?)"
pairs(fm1, ~ranef(.))
```

PBG

PBG

T5T4T3T2T1P5P3P2P4P1
MDL 72222Placebo
53241

pdBlocked

pdBlockedpdMatvaluenumeric(0)pdMatobjectpdBlockedcoefmatrixvaluepdMatobject
as.matrixpdMatvaluevalue

pdBlocked(value, form, nam, data, pdClass)

value	valuepdMatpdMatnumeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors
pdClass	pdMatvaluepdMat"pdSymm"

pdBlockedpdMat

<bates@stat.wisc.edu>

```
as.matrix.pdMatcoef.pdMatpdClassesmatrix<-pdMat
```

```
pd1 <- pdBlocked(list(diag(1:2), diag(c(0.1, 0.2, 0.3))),  
                  nam = list(c("A","B"), c("a1", "a2", "a3")))  
pd1
```

pdClasses

pdMat

pdSymm

pdLogChol

pdDiag

pdIdent

pdCompSymm

pdBlocked pdMat

pdNatural

pdMatpdConstructpdMatrixcoefpdSymmpdDiag

<bates@stat.wisc.edu>

pdBlockedpdCompSymmpdDiagpdFactorpdIdentpdMatpdMatrixpdNaturalpdSymmpdLogChol

pdCompSymm

pdCompSymmvaluenumeric(0)pdMatobjectpdCompSymmcoefmatrixvaluepdMatobject
as.matrix(value)value

pdCompSymm(value, form, nam, data)

value	pdMat+numeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors

pdCompSymmpdMat

<bates@stat.wisc.edu>

[as.matrix.pdMatcoef.pdMatmatrix<-.pdMatpdClasses](#)

```
pd1 <- pdCompSymm(diag(3) + 1, nam = c("A","B","C"))  
pd1
```

pdConstruct

pdMatobjectpdMat

pdConstruct(object, value, form, nam, data, ...)

object	pdMat
value	pdMat+numeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors
...	

pdMatobject

<bates@stat.wisc.edu>

[pdCompSymmpdDiagpdIdentpdNaturalpdSymm](#)

```
pd1 <- pdSymm()
pdConstruct(pd1, diag(1:4))
```

pdConstruct.pdBlocked

```
pdBlockedpdMatvaluenumeric(0)pdMatobjectpdBlockedcoefmatrixvaluepdMatobject
as.matrixpdMatvaluevalue
```

```
## S3 method for class 'pdBlocked'
pdConstruct(object, value, form, nam, data, pdClass,
...)
```

```
object      "pdBlocked"
value       valuepdMatpdMatnumeric(0)
form        objectformvalueNULL
nam         valueNULL
data        valueformfactorsNULLfactors
pdClass     pdMatvaluepdMat"pdSymm"
...
```

pdBlockedpdMat

<bates@stat.wisc.edu>

[as.matrix.pdMatcoef.pdMatpdBlockedpdClassespdConstructmatrix<-pdMat](#)

```
pd1 <- pdBlocked(list(c("A","B"), c("a1", "a2", "a3")))
pdConstruct(pd1, list(diag(1:2), diag(c(0.1, 0.2, 0.3))))
```

pdDiag

pdDiagobject nn valuenumeric(0)pdMatobjectpdDiagcoefmatrixvaluepdMatobject
as.matrix(value)value

pdDiag(value, form, nam, data)

value	pdMat+numeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors

pdDiagpdMat

<bates@stat.wisc.edu>

[as.matrix.pdMatcoef.pdMatpdClassesmatrix<-pdMat](#)

```
pd1 <- pdDiag(diag(1:3), nam = c("A", "B", "C"))
pd1
```

pdFactor

object $\Sigma\Sigma L\Sigma = L'LL$

pdFactor(object)

objectpdMatlength(coef(object)) > 0

object

pdMatrix

<bates@stat.wisc.edu>

pdMatrix

```
pd1 <- pdCompSymm(4 * diag(3) + 1)
pdFactor(pd1)
```

pdFactor.reStruct

pdMatobject

```
## S3 method for class 'reStruct'
pdFactor(object)
```

```
object          "reStruct"pdMat
```

object

pdMatrix

<bates@stat.wisc.edu>

pdFactorpdMatrix.reStructpdFactor.pdMat

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))
pdFactor(rs1)
```

pdIdent

pdIdentobjectvaluenumeric(0)pdMatobjectpdIdentcoefmatrixvaluepdMatobject
as.matrix(value)value

pdIdent(value, form, nam, data)

value	pdMat+numeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors

pdIdentpdMat

<bates@stat.wisc.edu>

[as.matrix.pdMatcoef.pdMatpdClassesmatrix<-.pdMat](#)

```
pd1 <- pdIdent(4 * diag(3), nam = c("A", "B", "C"))  
pd1
```

pdLogChol

pdLogCholobjectnn($n + 1$)/2
valuenumeric(0)pdMatobjectpdLogCholcoefmatrix
valuepdMatobjectas.matrix(value)
value

pdLogChol(value, form, nam, data)

```
value      pdMat+numeric(0)
form       objectformvalueNULL
nam        valueNULL
data       valueformfactorsNULLfactor
```

```
pdLogChol
```

```
pdLogCholpdMat
```

```
<bates@stat.wisc.edu>
```

```
as.matrix.pdMatcoef.pdMatpdClassesmatrix<-.pdMat
```

```
(pd1 <- pdLogChol(diag(1:3), nam = c("A","B","C")))  
  
(pd4 <- pdLogChol(1:6))  
(pd4c <- chol(pd4)) # -> upper-tri matrix with off-diagonals  4 5 6  
pd4c[upper.tri(pd4c)]  
log(diag(pd4c)) # 1 2 3
```

```
pdMat
```

```
pdMatpdClassdata.class(object)objectpdMatpdMat
```

```
pdMat(value, form, nam, data, pdClass)
```

```
value      pdMat+numeric(0)
form       objectformvalueNULL
nam        valueNULL
data       valueformfactorsNULLfactors
pdClass    pdMatvaluepdMat"pdSymm"
```



```
pdMatpdClassclass(object)objectpdMat
```

```
<bates@stat.wisc.edu>
```

[pdClassespdCompSymmpdDiagpdIdentpdNaturalpdSymmreStructsolve.pdMatsummary.pdMat](#)

```
pd1 <- pdMat(diag(1:4), pdClass = "pdDiag")
pd1
str(pd1)
```

pdMatrix

```
object $\Sigma\Sigma L\Sigma = L' L\Sigma L$ 
```

```
pdMatrix(object, factor)
```

object	pdMat
factor	TRUEobjectFALSEFALSE

```
factorFALSEobject
```

```
<bates@stat.wisc.edu>
```

[as.matrix.pdMatpdClassespdFactorpdMatpdMatrix.reStructcorMatrix](#)

```
pd1 <- pdSymm(diag(1:4))
pdMatrix(pd1)
```

pdMatrix.reStruct

pdMatobject

```
## S3 method for class 'reStruct'  
pdMatrix(object, factor)
```

```
object      "reStruct"pdMat  
factor      TRUEobjectFALSEFALSE
```

object

<bates@stat.wisc.edu>

[as.matrix.reStructreStructpdMatpdMatrixpdMatrix.pdMat](#)

```
rs1 <- reStruct(pdSymm(diag(3), ~age+Sex, data = Orthodont))  
pdMatrix(rs1)
```

pdNatural

pdNaturalobject $n(n+1)/2\sigma_{ij}i,j\rho_{ij} = \sigma_i/\sqrt{\sigma_{ii}\sigma_{jj}}, i \neq j\sqrt{\sigma_{ii}}, i = 1, \dots, n\log((1 + \rho_{ij})/(1 - \rho_{ij})), i \neq j$ valuenumeric(0)pdMatobjectpdSymmcoefmatrixvaluepdMatobject
as.matrix(value)value

pdNatural(value, form, nam, data)

```
value      pdMat+numeric(0)  
form       objectformvalueNULL  
nam        valueNULL  
data       valueformfactorsNULLfactors
```

pdNaturalpdMat

<bates@stat.wisc.edu>

as.matrix.pdMatcoef.pdMatpdClassesmatrix<- .pdMat

pdNatural(diag(1:3))

pdSymm

pdSymmobjectnn($n + 1$)/2valuenumeric(0)pdMatobjectpdSymmcoefmatrixvaluepdMat
objectas.matrix(value)value

pdSymm(value, form, nam, data)

value	pdMat+numeric(0)
form	objectformvalueNULL
nam	valueNULL
data	valueformfactorsNULLfactors

pdSymmpdMat

<bates@stat.wisc.edu>

as.matrix.pdMatcoef.pdMatpdClassesmatrix<- .pdMat

pd1 <- pdSymm(diag(1:3), nam = c("A", "B", "C"))
pd1

Phenobarb

Phenobarb

< 5>= 5

u

u

phenoModel

Phenobarb

phenoModel(Subject, time, dose, lCl, lV)

Subject

time

dose *u*

lCl Subjecttime

lV Subjecttime

[Phenobarb](#)phenoModel

<bates@stat.wisc.edu>

Pixel

Pixel

110

LR

```
fm1 <- lme(pixel ~ day + I(day^2), data = Pixel,
            random = list(Dog = ~ day, Side = ~ 1))
summary(fm1)
VarCorr(fm1)
```

plot.ACF

xyplottype = "h"alpha > 0alpha

```
## S3 method for class 'ACF'
plot(x, alpha, xlab, ylab, grid, ...)
```

x	ACFlagACF
alpha	alpha0
xlabylab	"Lag""Autocorrelation"
grid	FALSE
...	xyplot

xyplot

<bates@stat.wisc.edu>

[ACFxyplot](#)

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
plot(ACF(fm1, maxLag = 10), alpha = 0.01)
```

plot.augPred

xyplot

```
## S3 method for class 'augPred'
plot(x, key, grid, ...)
```

x	" augPred "
key	TRUEFALSEkeytrellisxyplotTRUE
grid	FALSE
...	trellis

<bates@stat.wisc.edu>

[augPredxyplot](#)

```
fm1 <- lme(Orthodont)
plot(augPred(fm1, level = 0:1, length.out = 2))
```

plot.compareFits

dotplot

```
## S3 method for class 'compareFits'
plot(x, subset, key, mark, ...)
```

x	"compareFits"
subset	x
key	TRUEFALSEkeytrellisdotplotTRUE
mark	
...	trellis

dotplot

<bates@stat.wisc.edu>

[compareFitspairs.compareFitsdotplot](#)

```
example(compareFits) # cF12 <- compareFits(coef(lmList(Orthodont)), .. lme(*))
plot(cF12)
```

plot.gls

form|form|histogramformxyplotformbwplot

```
## S3 method for class 'glms'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
```

x	"glms"
form	xx". " resid(., type = "p") ~ fitted(.)
abline	
id	1 - value/2idLabels
idLabels	id
idResType	id"pearson""normalized""pearson"
grid	xyplotTRUEFALSE
...	

<bates@stat.wisc.edu>

[glscopyplotbwplot](#)[histogram](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
# standardized residuals versus fitted values by Mare
plot(fm1, resid(., type = "p") ~ fitted(.) | Mare, abline = 0)
# box-plots of residuals by Mare
plot(fm1, Mare ~ resid(.))
# observed versus fitted values by Mare
plot(fm1, follicles ~ fitted(.) | Mare, abline = c(0,1))
```

`plot.intervals.lmList`

```
lm1lmListx"+"
dotplot()

## S3 method for class 'intervals.lmList'
plot(x, xlab = "", ylab = attr(x, "groupsName"),
     strip = function(...) strip.default(..., style = 1),
     ...)

x           "intervals.lmList"lm1lmListx
xlabylab
strip       functionFALSEdotplot()
...         dotplot

lm1lmListx
```

<bates@stat.wisc.edu>

[intervals.lmList](#)[lmList](#)[dotplot](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
plot(intervals(fm1))
```

plot.lme

form|form|histogramformxyplotformbwplot

```
## S3 method for class 'lme'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
## S3 method for class 'nls'
plot(x, form, abline, id, idLabels, idResType, grid, ...)
```

```
x          "lme"nls
form       xx". "|resid(., type = "p") ~ fitted(.)
abline
id         1 - value/2idLabels
idLabels   id
idResType  id"pearson""normalized""pearson"
grid       xyplotTRUEFALSE
...
```

<bates@stat.wisc.edu>

[lmexyplotbwplot](#)[histogram](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
# standardized residuals versus fitted values by gender
plot(fm1, resid(., type = "p") ~ fitted(.) | Sex, abline = 0)
# box-plots of residuals by Subject
plot(fm1, Subject ~ resid(.))
# observed versus fitted values by Subject
plot(fm1, distance ~ fitted(.) | Subject, abline = c(0,1))
```

plot.lmList

xform|form|histogramformmxyplotformbwplot

```
## S3 method for class 'lmList'
plot(x, form, abline, id, idLabels, grid, ...)
```

```
x          "lmList"lm
form       xx". "|resid(., type = "pool") ~ fitted(.)
abline
id         1 - value/2idLabels
idLabels   idgetGroups(x)
grid       xyplotTRUEFALSE
...
```

<bates@stat.wisc.edu>

[lmListpredict.lmxyplotbwplot](#)[histogram](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
# standardized residuals versus fitted values by gender
plot(fm1, resid(., type = "pool") ~ fitted(.) | Sex, abline = 0, id = 0.05)
# box-plots of residuals by Subject
plot(fm1, Subject ~ resid(.))
# observed versus fitted values by Subject
plot(fm1, distance ~ fitted(.) | Subject, abline = c(0,1))
```

plot.nffGroupedData

dotplot

```
## S3 method for class 'nffGroupedData'
plot(x, outer, inner, innerGroups, xlab, ylab, strip, panel, key,
     grid, ...)
```

x	nfnGroupedData
outer	TRUEattr(object, "outer")NULL
inner	TRUEattr(object, "inner")NULL
innerGroups	innerinnerGroupsNULLinnerGroups
xlab	yattr(object, "labels")attr(object, "units")
ylab	
strip	stripdotplotstrip.default(..., style = 1)trellis.args
panel	paneldotplot
key	TRUEinnerinnerGroupsNULLinnerinnerGroupskeydotplotTRUEinnerinnerGroupsNULLFALSE
grid	plot.nfnGroupedData
...	dotplot

<bates@stat.wisc.edu>

[groupedDatadotplot](#)

```
plot(Machines)
plot(Machines, inner = TRUE)
```

plot.nfnGroupedData

xyplot

```
## S3 method for class 'nfnGroupedData'
plot(x, outer, inner, innerGroups, xlab, ylab, strip, aspect, panel,
      key, grid, ...)
```

x	nfnGroupedDatagroupedData
outer	TRUEattr(object, "outer")NULL
inner	TRUEattr(object, "inner")NULL
innerGroups	innerinnerGroupsNULLinnerGroups
xlaby	attr(object, "labels")attr(object, "units")
strip	stripxyplotstrip.default(..., style = 1)trellis.args
aspect	aspectxyplot"xy"trellis.args
panel	panelxyplot
key	TRUEinnerGroupsNULLinnerGroupskeyxyplotTRUEinnerGroupsNULLFALSE
grid	TRUE
...	xyplot

<bates@stat.wisc.edu>

[groupedData](#)[xyplot](#)

```
# different panels per Subject
plot(Orthodont)
# different panels per gender
plot(Orthodont, outer = TRUE)
```

plot.nmGroupedData

groupedDatadisplayLevelpreservegroupedDatadisplayLevelplotgroupedData

```
## S3 method for class 'nmGroupedData'
plot(x, collapseLevel, displayLevel, outer, inner,
     preserve, FUN, subset, key, grid, ...)
```

x	nmGroupedDatagroupedData
collapseLevel	
displayLevel	outercollapseLevel
outer	displayLevelTRUEdisplayLevelattr(object, "outer")NULL
inner	displayLevelTRUEattr(object, "outer")NULL
preserve	collapseLevelcollapseLevelNULL
FUN	objectcollapseLevelFUNFUNOrderedfactornumericmeanModedefactor orderedModegsummarymode
subset	NULL
key	TRUEFALSEkeytrellisxyplotTRUE
grid	TRUE
...	

collapseLeveldisplayLevel

<bates@stat.wisc.edu>

[groupedDatacollapse.groupedDataplot.nfnGroupedDataplot.nffGroupedData](#)

```
# no collapsing, panels by Dog
plot(Pixel, displayLevel = "Dog", inner = ~Side)
# collapsing by Dog, preserving day
plot(Pixel, collapseLevel = "Dog", preserve = ~day)
```

plot.ranef.lme

"Trellis"[ranef\(lme\(*\)\)](#)"[ranef.lme](#)"

```
## S3 method for class 'ranef.lme'
plot(x, form = NULL, omitFixed = TRUE, level = Q,
     grid = TRUE, control, xlab, ylab, strip,
     ...)
```

```

x          "ranef.lme"lme
form

          dotplot()form
          NULLxyplot()NULL

omitFixed  TRUE
level      xx
grid       formTRUE
control    formcontroldrawLineIoessTRUEspan.loessspanpanel.loess2/3
           degree.loessdegreepanel.loess1cex.axis0.8srt.axis0mgp.axis
           c(2, 0.5, 0)

xlabylab
strip      functionFALSEdotplot()
...        dotplot

formdotplot()form~g~g1*g2form
formx+xyplot()formfactorordered

```

<bates@stat.wisc.edu>

[ranef.lmelmidotplot](#)

```

fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
plot(ranef(fm1))
fm1RE <- ranef(fm1, augFrame = TRUE)
plot(fm1RE, form = ~ Sex)
plot(fm1RE, form = age ~ Sex) # "connected" boxplots

```

plot.ranef.lmList

```

formformformformdotplot
formformfactorordered

```

```

## S3 method for class 'ranef.lmList'
plot(x, form, grid, control, ...)

```

```

x          "ranef.lmList"lmList
form       dotplotform~g~g1*g2x+NULL
grid       formFALSE
control    formcontroldrawLineIoessTRUEspan.loessspanpanel.loess2/3
           degree.loessdegreepanel.loess1cex.axis0.8srt.axis0mgp.axis
           c(2, 0.5, 0)
...        dotplot

```

<bates@stat.wisc.edu>

lmListdotplot

```

fm1 <- lmList(distance ~ age | Subject, Orthodont)
plot(ranef(fm1))
fm1RE <- ranef(fm1, augFrame = TRUE)
plot(fm1RE, form = ~ Sex)
plot(fm1RE, form = age ~ Sex)

```

plot.Variogram

```

xyplotsmooth = TRUEloessshowModel = TRUEx"modelVariog"

```

```

## S3 method for class 'Variogram'
plot(x, smooth, showModel, sigma, span, xlab,
      ylab, type, ylim, grid, ...)

```

```

x          "Variogram"variogdist
smooth     loessTRUEshowModelFALSE
showModel  "modelVariog"xTRUE"modelVariog"
sigma      NULL
span       loess
xlabylab   "Distance""SemiVariogram"
type       "p"
ylim       c(0, max(x$variog))
grid       FALSE
...        xyplot

```

xyplot

<bates@stat.wisc.edu>

Variogramxyplotloess

```
fm1 <- lme(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary)
plot(Variogram(fm1, form = ~ Time | Mare, maxDist = 0.7))
```

pooledSD

object

pooledSD(object)

object lmList

objectdf

<bates@stat.wisc.edu>

lmListlm

```
fm1 <- lmList(Orthodont)
pooledSD(fm1)
```

predict.gls

objectnewdata

```
## S3 method for class 'glS'  
predict(object, newdata, na.action, ...)
```

```
object          "glS"  
newdata  
na.action       newdataNAna.fail  
...
```

<bates@stat.wisc.edu>

glS

```
fm1 <- glS(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,  
           correlation = corAR1(form = ~ 1 | Mare))  
newOvary <- data.frame(Time = c(-0.75, -0.5, 0, 0.5, 0.75))  
predict(fm1, newOvary)
```

predict.gnlS

objectnewdata

```
## S3 method for class 'gnlS'  
predict(object, newdata, na.action, naPattern, ...)
```

```
object          "gnlS"  
newdata  
na.action       newdataNAna.fail  
naPattern  
...
```

<bates@stat.wisc.edu>

[gnls](#)

```
fm1 <- gnls(weight ~ SSlogis(Time, Asym, xmid, scal), Soybean,
            weights = varPower())
newSoybean <- data.frame(Time = c(10,30,50,80,100))
predict(fm1, newSoybean)
```

`predict.lme`

newdataNA

```
## S3 method for class 'lme'
predict(object, newdata, level = Q, asList = FALSE,
        na.action = na.fail, ...)
```

```
object          "lme"
newdata
level
asList          TRUElevellevel
na.action       newdataNA na.fail
...
```

```
levelasList = TRUEasList = FALSElevel
```

<bates@stat.wisc.edu>

[lme](#)
[fitted.lme](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
newOrth <- data.frame(Sex = c("Male","Male","Female","Female","Male","Male"),
                     age = c(15, 20, 10, 12, 2, 4),
                     Subject = c("M01","M01","F30","F30","M04","M04"))
## The 'Orthodont' data has *no* 'F30', so predict NA at level 1 :
predict(fm1, newOrth, level = 0:1)
```

`predict.lmList`

```
objectnewdataobjectnewdataobjectse.fit=TRUE
```

```
## S3 method for class 'lmList'
predict(object, newdata, subset, pool, asList, se.fit, ...)
```

```
object      "lmList"lm
newdata     objectobject
subset      lmobjectNULL
asList      TRUEFALSE
pool        attr(object, "pool")
se.fit      FALSE
...
```

```
lmobjectlmobject
```

```
<bates@stat.wisc.edu>
```

```
lmListpredict.lm
```

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
predict(fm1, se.fit = TRUE)
```

`predict.nlme`

```
iiinewdataNA
```

```
## S3 method for class 'nlme'
predict(object, newdata, level = Q, asList = FALSE,
        na.action = na.fail, naPattern = NULL, ...)
```

```

object      "nlme"
newdata
level       object$dim$Q
asList      TRUElevellevel
na.action   newdataNAfail
naPattern
...

```

```
levelasList = TRUEasList = FALSElevel
```

```
<bates@stat.wisc.edu>
```

```
nlmefitted.lme
```

```

head(Loblolly) # groupedData w/ 'Seed' is grouping variable :
## Grouped Data: height ~ age | Seed
##   height age Seed
## 1    4.51  3  301
## 15  10.89  5  301
## .. ..... . ...

fm1 <- nlme(height ~ SSasym(age, Asym, R0, lrc), data = Loblolly,
            fixed = Asym + R0 + lrc ~ 1,
            random = Asym ~ 1, ## <---grouping---> Asym ~ 1 | Seed
            start = c(Asym = 103, R0 = -8.5, lrc = -3.3))
fm1

age. <- seq(from = 2, to = 30, by = 2)
newLL.301 <- data.frame(age = age., Seed = 301)
newLL.329 <- data.frame(age = age., Seed = 329)
(p301 <- predict(fm1, newLL.301, level = 0:1))
(p329 <- predict(fm1, newLL.329, level = 0:1))
## Prediction are the same at level 0 :
all.equal(p301[, "predict.fixed"],
          p329[, "predict.fixed"])
## and differ by the 'Seed' effect at level 1 :
p301[, "predict.Seed"] -
p329[, "predict.Seed"]

```

```
print.summary.pdMat
```

```
objectobject
```

```
## S3 method for class 'summary.pdMat'  
print(x, sigma, rdig, Level, resid, ...)
```

```
x          "summary.pdMat"summary"pdMat"  
sigma      object  
rdig  
Level      object  
resid      TRUE"residual"sigmaFALSE  
...        print.default
```

```
<bates@stat.wisc.edu>
```

```
summary.pdMatpdMat
```

```
pd1 <- pdCompSymm(3 * diag(2) + 1, form = ~age + age^2,  
  data = Orthodont)  
print(summary(pd1), sigma = 1.2, resid = TRUE)
```

```
print.varFunc
```

```
x
```

```
## S3 method for class 'varFunc'  
print(x, ...)
```

```
x          "varFunc"  
...        print.default
```

```
<bates@stat.wisc.edu>
```

```
summary.varFunc
```

```
vf1 <- varPower(0.3, form = ~age)  
vf1 <- Initialize(vf1, Orthodont)  
print(vf1)
```

`qqnorm.gls`

`form|`

```
## S3 method for class 'gls'
qqnorm(y, form, abline, id, idLabels, grid, ...)
```

```
y          "gls"
form        yy"."|form|~ resid(., type = "p")
abline
id          1 - value/2idLabels
idLabels    id
grid        xyplotTRUEFALSE
...
```

`<bates@stat.wisc.edu>``glspplot.gls`

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
qqnorm(fm1, abline = c(0,1))
```

`qqnorm.lme`

`form|`

```
## S3 method for class 'lme'
qqnorm(y, form, abline, id, idLabels, grid, ...)
```

```

y          "lme""lmList"lm"lm""nls"
form       yy"."|form|~ resid(., type = "p")
abline
id         1 - value/2idLabels
idLabels   id
grid       FALSE
...

```

<bates@stat.wisc.edu>

[lmeplot.lme](#)

```

fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
## normal plot of standardized residuals by gender
qqnorm(fm1, ~ resid(., type = "p") | Sex, abline = c(0, 1))
## normal plots of random effects
qqnorm(fm1, ~ranef(.))

```

Quinidine

Quinidine

CaucasianLatinBlack
 noyes
 nonecurrentformer
 No/MildModerateSevere
 < 50>= 50

quinModel

Quinidine

quinModel(Subject, time, conc, dose, interval, lV, lKa, lCl)

Subject

time

conc

dose

interval

lV Subjecttime

lKa Subjecttime

lCl Subjecttime

[Quinidine](#)quinModel

<bates@stat.wisc.edu>

Rail

Rail

random.effects

lmListlme

random.effectsranef

random.effects(object, ...)

ranef(object, ...)

object

...

[ranef.lmListranef.lme](#)

ranef.lme

i

```
## S3 method for class 'lme'
ranef(object, augFrame, level, data, which, FUN,
       standard, omitGroupingFactor, subset, ...)
```

```
object          "lme"
augFrame        TRUEdataFALSEFALSE
level
data            augFrame = TRUEobject
which           datadata
FUN             dataFUNFUNorderedfactornumericmeanModedefactororderedModegsummary
               mode
standard        FALSE
omitGroupingFactor
               TRUEdataFALSE
subset
...
```

```
levelrandom.effects.lmedata.frame
```

```
<bates@stat.wisc.edu>
```

[coef.lmegsummarylmeplot.ranef.lmerandom.effects](#)

```
fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
ranef(fm1)
random.effects(fm1)          # same as above
random.effects(fm1, augFrame = TRUE)
```

ranef.lmList

lm

```
## S3 method for class 'lmList'
ranef(object, augFrame, data, which, FUN, standard,
       omitGroupingFactor, ...)
```

```
object          "lmList"lm
augFrame         TRUEdataFALSEFALSE
data            augFrame = TRUEobject
which           datadata
FUN             dataFUNFUNorderedfactornumericmeanModedefactororderedModegsummary
               mode
standard        FALSE
omitGroupingFactor
               TRUEdataFALSE
...
```

lmobject

<bates@stat.wisc.edu>

[fixed.effects.lmListlmListrandom.effects](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
ranef(fm1)
random.effects(fm1)          # same as above
```

RatPupWeight

RatPupWeight

MaleFemale
987421013562122242726252317111413151620191812

ControlLowHigh

recalc

corStructmodelStructreStructvarFunc

recalc(object, conLin, ...)

object	conLin
conLin	"Xy"Xy"logLik"
...	

lmegls

<bates@stat.wisc.edu>

[recalc.corStructrecalc.modelStructrecalc.reStructrecalc.varFunc](#)

recalc.corStruct

```
"Xy"conLinobjectobjectlogLik(object)"logLik"conLin
```

```
## S3 method for class 'corStruct'  
recalc(object, conLin, ...)
```

```
object      "corStruct"  
conLin      "Xy"Xy"logLik"  
...
```

lmegls

<bates@stat.wisc.edu>

[corFactorlogLik.corStruct](#)

recalc.modelStruct

```
object
```

```
## S3 method for class 'modelStruct'  
recalc(object, conLin, ...)
```

```
object      "modelStruct"corStructvarFunc  
conLin      "Xy"Xy"logLik"attr(object, "conLin")  
...
```

lmegls

<bates@stat.wisc.edu>

[recalc.corStructrecalc.reStructrecalc.varFunc](#)

recalc.reStruct

objectconLincoef(object)logLikconLinsettingsobject

S3 method for class 'reStruct'
recalc(object, conLin, ...)

object "reStruct"pdMat
conLin "Xy"Xy"logLik"
...

logLik

<bates@stat.wisc.edu>

[logLiklmerecalcreStruct](#)

recalc.varFunc

"Xy"conLinobjectobjectlogLik(object)"logLik"conLin

S3 method for class 'varFunc'
recalc(object, conLin, ...)

object "varFunc"
conLin "Xy"Xy"logLik"
...

lmegls

<bates@stat.wisc.edu>

[recalcvarWeightslogLik.varFunc](#)

Relaxin

Relaxin

589342716

Remifentanil

Remifentanil

Remifentanil

"groupedData""nfnGroupedData"

ID

Subject `ordered`ID30212523293665109

Time `numeric`

conc

Rate

Amt

Age

Sex `factor`FemaleMale

Ht

Wt

BSA $BSA := Wt^{0.425} \cdot Ht^{0.725} \cdot 0.007184$

LBM $LBM_m := 1.1Wt - 128(Wt/Ht)^2$ $LBM_f := 1.07Wt - 148(Wt/Ht)^2$

```

plot(Remifentanil, type = "l", lwd = 2) # shows the 65 patients' remi profiles

## The same on log-log scale (*more* sensible for modeling?):
plot(Remifentanil, type = "l", lwd = 2, scales = list(log=TRUE))

str(Remifentanil)
summary(Remifentanil)

plot(xtabs(~Subject, Remifentanil))
summary(unclass(table(Remifentanil$Subject)))
## between 20 and 54 measurements per patient (median: 24; mean: 32.42)

## Only first measurement of each patient :
dim(Remi.1 <- Remifentanil[!duplicated(Remifentanil[, "ID"]),]) # 65 x 12

LBMfn <- function(Wt, Ht, Sex) ifelse(Sex == "Female",
                                     1.07 * Wt - 148*(Wt/Ht)^2,
                                     1.1 * Wt - 128*(Wt/Ht)^2)

with(Remi.1,
     stopifnot(all.equal(BSA, Wt^{0.425} * Ht^{0.725} * 0.007184,
                        tolerance = 1.5e-5),
               all.equal(LBM, LBMfn(Wt, Ht, Sex),
                        tolerance = 7e-7))
))

## Rate: typically 3 µg / kg body weight, but :
sunflowerplot(Rate ~ Wt, Remifentanil)
abline(0,3, lty=2, col=adjustcolor("black", 0.5))

```

residuals.gls

object

```

## S3 method for class 'glms'
residuals(object, type, ...)

```

```

object      "glms"gnls
type        "response""pearson""normalized""response"
...

```

object

<bates@stat.wisc.edu>

gls

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,  
           correlation = corAR1(form = ~ 1 | Mare))  
residuals(fm1)
```

residuals.glsStruct

object

```
## S3 method for class 'glsStruct'  
residuals(object, glsFit, ...)
```

```
object      "glsStruct"corStruct"varFunc"  
glsFit      logLikbetasigmavarBetafittedresidualsattr(object, "glsFit")  
...
```

object

glsresiduals.gls

<bates@stat.wisc.edu>

glsStructresiduals.glsfitted.glsStruct

`residuals.gnlsStruct`

`object`

```
## S3 method for class 'gnlsStruct'  
residuals(object, ...)
```

```
object          "gnlsStruct"corStructvarFunc  
...
```

`object``gnlsresiduals.gnls``<bates@stat.wisc.edu>``gnlsresiduals.gnlsfitted.gnlsStruct`

`residuals.lme`

`itype="pearson"`

```
## S3 method for class 'lme'  
residuals(object, level = Q,  
           type = c("response", "pearson", "normalized"), asList = FALSE, ...)
```

```
object          "lme"  
level           object  
type            "response""pearson""normalized"  
asList          TRUElevellevelFALSE  
...
```

`levelasList = TRUEasList = FALSElevelnaresid`

<bates@stat.wisc.edu>

`lme`
`fitted.lme`

```
fm1 <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1)
head(residuals(fm1, level = 0:1))
summary(residuals(fm1) /
        residuals(fm1, type = "p")) # constant scaling factor 1.432
```

`residuals.lmeStruct`

iii

```
## S3 method for class 'lmeStruct'
residuals(object, level, conLin, lmeFit, ...)
```

object	" <code>lmeStruct</code> "reStructcorStructvarFunc
level	object
conLin	"Xy"Xy"logLik"attr(object, "conLin")
lmeFit	betabattr(object, "lmeFit")
...	

levellevel

`lme`

<bates@stat.wisc.edu>

`lmeresiduals.lme`
`fitted.lmeStruct`

`residuals.lmList`

`lmobjectobject`

```
## S3 method for class 'lmList'
residuals(object, type, subset, asList, ...)
```

```
object      "lmList"lm
subset      lmobjectNULL
type        "response""pearson"lm"pooled.pearson""response"
asList      TRUEFALSE
...
```

`lmobjectlmobject``<bates@stat.wisc.edu>``lmListfitted.lmList`

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
residuals(fm1)
```

`residuals.nlmeStruct`

`iii`

```
## S3 method for class 'nlmeStruct'
residuals(object, level, conLin, ...)
```

```
object      "nlmeStruct"reStructcorStructvarFunc
level       object
conLin      "Xy"Xy"logLik"attr(object, "conLin")
...
```

levellevel

nlme

<bates@stat.wisc.edu>

[nlmefitted.nlmeStruct](#)

reStruct

reStructpdMatsettings

```
reStruct(object, pdClass, REML, data)
## S3 method for class 'reStruct'
print(x, sigma, reEstimates, verbose, ...)
```

object	$\sim x_1 + \dots + x_n \mid g_1 / \dots / g_m x_1 + \dots + x_n g_1 / \dots / g_m m / \sim x_1 + \dots + x_n \mid g$ $\sim x_1 + \dots + x_n$ pdMatNULLformula(object)pdMatgroupedDatapdMatreStruct
pdClass	pdMatobject"pdLogChol"
REML	TRUEFALSEFALSE
data	objectfactorspdMatNULLfactors
x	reStruct
sigma	pdMat
reEstimates	verbose = TRUE
verbose	FALSE
...	

reStruct

<bates@stat.wisc.edu>

[groupedDataImepdMatsolve.reStructsummary.reStructupdate.reStruct](#)

```
rs1 <- reStruct(list(Dog = ~day, Side = ~1), data = Pixel)
rs1 # 2 entries "Uninitialized"
str(rs1) # a bit more
```

simulate.lme	lme
--------------	-----

```
objectnsimobjectm2
```

```
## S3 method for class 'lme'
simulate(object, nsim = 1, seed = , m2,
         method = c("REML", "ML"), niterEM = c(40, 200), useGen, ...)
```

```
object      "lme"lmefixeddatarandomlme
m2          "lme"objectobjectm2
seed        set.seed
method      "REML""ML"c("REML", "ML")
nsim        1
niterEM     objectm2c(40,200)
useGen      TRUEenlminbFALSEnlm"pdMat"objectm2pdClassesFALSETRUE
...
```

```
simulate.lmenullaltMLREMLseed
```

```
<bates@stat.wisc.edu>
```

```
lmeset.seed
```

```
orthSim <-
  simulate.lme(list(fixed = distance ~ age, data = Orthodont,
                   random = ~ 1 | Subject),
               nsim = 3, # limited here for speed
               m2 = list(random = ~ age | Subject))
```

solve.pdMat

aa

```
## S3 method for class 'pdMat'  
solve(a, b, ...)
```

```
a          "pdMat"  
b  
...
```

pdMataa

<bates@stat.wisc.edu>

pdMat

```
pd1 <- pdCompSymm(3 * diag(3) + 1)  
solve(pd1)
```

solve.reStruct

SolvepdMata

```
## S3 method for class 'reStruct'  
solve(a, b, ...)
```

```
a          "reStruct"pdMat  
b  
...
```

reStructapdMata

<bates@stat.wisc.edu>

solve.pdMatreStruct

```
rs1 <- reStruct(list(A = pdSymm(diag(1:3), form = ~Score),  
  B = pdDiag(2 * diag(4), form = ~Educ)))  
solve(rs1)
```

Soybean

Soybean

```
summary(fm1 <- nlsList(SSlogis, data = Soybean))
```

splitFormula

formsepformsepform

splitFormula(form, sep)

form	formula
sep	"/"

formsep

<bates@stat.wisc.edu>

[formula](#)

```
splitFormula(~ g1/g2/g3)
```

Spruce

Spruce

summary.corStruct

objectprint.summarystructName

```
## S3 method for class 'corStruct'
summary(object, structName, ...)
```

```
object      "corStruct"
structName  objectprint.summaryclass(object)[1]
...
```

objectsummary.corStructstructNameobject

[corClassescorNaturalInitialize.corStructsummary](#)

```
cs1 <- corAR1(0.2)
summary(cs1)
```

summary.gls	gls
-------------	-----

objectobject

```
## S3 method for class 'gls'
summary(object, verbose, ...)
```

```
object      "gls"
verbose     FALSE
...
```

```
summary.glsobjectglsObject
```

```
corBeta
tTable      ValueStd. Error t-value p-value t
residuals   gls
AIC          object
BIC          object
```

```
<bates@stat.wisc.edu>
```

[AICBICglssummary](#)

```
fm1 <- gls(follicles ~ sin(2*pi*Time) + cos(2*pi*Time), Ovary,
           correlation = corAR1(form = ~ 1 | Mare))
summary(fm1)
coef(summary(fm1)) # "the matrix"
```

summary.lme

objectobject[printcoef](#)tTtable

```
## S3 method for class 'lme'
summary(object, adjustSigma, verbose, ...)
## S3 method for class 'summary.lme'
print(x, verbose = FALSE, ...)
```

```

object      "lme"
adjustSigma TRUEobject $\sqrt{n_{obs}/(n_{obs} - n_{par})}$ TRUE
verbose     print.summary.lmeFALSE
...         print
x           "summary.lme"

```

```
summary.lmeobjectlmeObject
```

```

corFixed
tTable      ValueStd. ErrorDf t-valuep-value
residuals   lme
AIC          object
BIC          object

```

```
<bates@stat.wisc.edu>
```

[AICBIClme](#)

```

fm1 <- lme(distance ~ age, Orthodont, random = ~ age | Subject)
(s1 <- summary(fm1))
coef(s1) # the (coef | Std.E | t | P-v ) matrix

```

```
summary.lmList
```

```
summary.lmlmobjectprint.summary.lmList
```

```

## S3 method for class 'lmList'
summary(object, pool, ...)

```

```

object      "lmList"lm
pool        attr(object, "pool")
...

```

```
summary.lmobjectsummary.lmListvalue
```

```
call          lmListobject
coefficients   lmobject"Value""Std. Error""t value""Pr(>|t|)"
correlation    lmobjectlm
cov.unscaled   lmobjectlm
df             lm
df.residual    lm
fstatistics    lm
pool          pool
r.squared      lm
residuals      lm
RSE
sigma          lm
terms          lm
```

```
<bates@stat.wisc.edu>
```

[lmListsummary](#)

```
fm1 <- lmList(distance ~ age | Subject, Orthodont)
summary(fm1)
```

```
summary.modelStruct
```

```
summaryobject
```

```
## S3 method for class 'modelStruct'
summary(object, ...)
```

```
object          "modelStruct"reStructcorStructvarFunc
...
```

```
objectsummary.modelStructobject
```

```
<bates@stat.wisc.edu>
```

reStructsummary

```
lms1 <- lmeStruct(reStruct = reStruct(pdDiag(diag(2), ~age)),
  corStruct = corAR1(0.3))
summary(lms1)
```

summary.nlsList

```
summarynlsobjectprint.summary.nlsList
```

```
## S3 method for class 'nlsList'
summary(object, ...)
```

```
object      "nlsList"nls
...          summary.lmListpoolattr(object, "pool")
```

```
summaryobjectsummary.nlsListvalue
```

```
call      nlsListobject
parameters nlsobject"Value""Std. Error""t value""Pr(>|t|)"
correlation nlsobjectnls
cov.unscaled lmobjectnls
df          nls
df.residual nls
pool        pool
RSE
sigma       lm
```

```
<bates@stat.wisc.edu>
```

nlsListsummary

```
fm1 <- nlsList(SSasyp, Loblolly)
summary(fm1)
```

summary.pdMat

structName noCorrelation object summary.pdMat

```
## S3 method for class 'pdMat'
summary(object, structName, noCorrelation, ...)
```

```
object          "pdMat"
structName      pdMat "Blocked" pdBlocked "Compound Symmetry" pdCompSymm
                "Diagonal" pdDiag "Multiple of an Identity" pdIdent "General
                Positive-Definite, Natural Parametrization" pdNatural "General
                Positive-Definite" pdSymm data.class(object) pdMat
noCorrelation   print.summary.pdMat FALSE pdDiag pdIdent TRUE
...
```

object structName noCorrelation summary.pdMat

<bates@stat.wisc.edu>

[print.summary.pdMat](#) pdMat

```
summary(pdSymm(diag(4)))
```

summary.varFunc

structName object summary.varFunc

```
## S3 method for class 'varFunc'
summary(object, structName, ...)
```

```
object          "varFunc"
structName      varFunc
                varComb "Combination of variance functions"
                varConstPower "Constant plus power of variance covariate"
                varConstProp "Constant plus proportion of variance covariate"
                varExp "Exponential of variance covariate"
                varIdent "Different standard deviations per stratum"
                varPower "Power of variance covariate"
                varFunc data.class(object)
...
```

objectstructNamesummary.varFunc

<bates@stat.wisc.edu>

varClassesvarFunc

```
vf1 <- varPower(0.3, form = ~age)
vf1 <- Initialize(vf1, Orthodont)
summary(vf1)
```

Tetracycline1

Tetracycline1

53241

tetrachelrtetracyn

Tetracycline2

Tetracycline2

45213

Berkmycintetramycin

update.modelStruct

objectdata

```
## S3 method for class 'modelStruct'  
update(object, data, ...)
```

```
object      "modelStruct"corStructvarFunc  
data        object  
...
```

object

lmegls

<bates@stat.wisc.edu>

[reStruct](#)

update.varFunc

formula(object)"."

```
## S3 method for class 'varFunc'  
update(object, data, ...)
```

```
object      "varFunc"  
data        ". "  
...
```

formula(object)"."varFuncobjectobject

<bates@stat.wisc.edu>

[needUpdatecovariate<-varFunc](#)

varClasses

varFunc

varExp
varPower
varConstPower
varConstProp
varIdent
varFixed
varComb

varFunccoefcoef<-InitializevarPower

<bates@stat.wisc.edu>

[varComb](#)[varConstPower](#)[varConstProp](#)[varExp](#)[varFixed](#)[varIdent](#)[varPower](#)[summary.varFunc](#)

varComb

varCombvarFunc...

varComb(...)

... varFunc

varCombvarFunc

<bates@stat.wisc.edu>

[varClasses](#)[varWeights](#)[varCombcoef](#)[varComb](#)

vf1 <- varComb(varIdent(form = ~1|Sex), varPower())

varConstPower

$$\text{varConstPower} v \sigma^2(v) v \sigma^2(v) = (\theta_1 + |v|_2^{\theta_2})^2 \theta_1, \theta_2 \theta_1, \theta_2$$

varConstPower(const, power, form, fixed)

const	power	form	form	form	const	numeric(0)	object	
form			~ v ~ v g v g".	"fitted(.)resid(.)form*~ v g1 * g2 * g3~ fitted(.)				
fixed			const	power	form	fixed	fixed	NULL

varConstPowervarFunc

<bates@stat.wisc.edu>

[varClasses](#)[varWeights](#).[varFunc](#)[coef](#).[varConstPower](#)

vf1 <- varConstPower(1.2, 0.2, form = ~age|Sex)

varConstProp

$$\text{varConstProp} v \sigma^2(v) v \sigma^2(v) = a^2 + b^2 * v^2$$

varConstProp(const, prop, form, fixed)

const	prop	form	form	form	const	const	prop	
form			~ v ~ v g v g".	"fitted(.)resid(.)form*~ v g1 * g2 * g3~ fitted(.)				
fixed			const	power	form	fixed	fixed	NULL

varConstPropvarFunc

varFunc

varConstPower varConstProp()

varClasses varWeights. varFunc coef. varFunc

```
# Generate some synthetic data using the two-component error model and use
# different variance functions, also with fixed sigma in order to avoid
# overparameterisation in the case of a constant term in the variance function
times <- c(0, 1, 3, 7, 14, 28, 56, 120)
pred <- 100 * exp(- 0.03 * times)
sd_pred <- sqrt(3^2 + 0.07^2 * pred^2)
n_replicates <- 2

set.seed(123456)
syn_data <- data.frame(
  time = rep(times, each = n_replicates),
  value = rnorm(length(times) * n_replicates,
    rep(pred, each = n_replicates),
    rep(sd_pred, each = n_replicates)))
syn_data$value <- ifelse(syn_data$value < 0, NA, syn_data$value)

f_const <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3))
f_varPower <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  weights = varPower())
f_varConstPower <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  weights = varConstPower())
f_varConstPower_sf <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  control = list(sigma = 1),
  start = list(parent_0 = 100, lrc = -3),
  weights = varConstPower())
f_varConstProp <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
```

```

    start = list(parent_0 = 100, lrc = -3),
    weights = varConstProp())
f_varConstProp_sf <- gnls(value ~ SSasyp(time, 0, parent_0, lrc),
  data = syn_data, na.action = na.omit,
  start = list(parent_0 = 100, lrc = -3),
  control = list(sigma = 1),
  weights = varConstProp())

AIC(f_const, f_varPower, f_varConstPower, f_varConstPower_sf,
  f_varConstProp, f_varConstProp_sf)

# The error model parameters 3 and 0.07 are approximately recovered
intervals(f_varConstProp_sf)

```

VarCorr

`"lme"` `"nlme"`

```

VarCorr(x, sigma = 1, ...)
## S3 method for class 'lme'
VarCorr(x, sigma = x$sigma, rdig = 3, ...)

## S3 method for class 'pdMat'
VarCorr(x, sigma = 1, rdig = 3, ...)
## S3 method for class 'pdBlocked'
VarCorr(x, sigma = 1, rdig = 3, ...)

```

```

x          "lme"
sigma      x$sigma1class(x)
rdig       3
...

```

VarianceStdDevCorr

<bates@stat.wisc.edu>

`lmenlme`

```

fm1 <- lme(distance ~ age, data = Orthodont, random = ~age)
VarCorr(fm1)

```

varExp

$\text{varExp} v \sigma^2(v) v \sigma^2(v) = \exp(2\theta v) \theta \theta$

`varExp(value, form, fixed)`

value	Valueformvalueformformvaluenumeric(0)object
form	~ v~ v gvg". "fitted(.)resid(.)form*~ v g1 * g2 * g3~ fitted(.)
fixed	formfixedfixedNULL

`varExpvarFunc`

<bates@stat.wisc.edu>

[varClassesvarWeights.varFunccoef.varExp](#)

```
vf1 <- varExp(0.2, form = ~age|Sex)
```

varFixed

$\text{varFixed} v \text{value} \sigma^2(v) \sigma^2(v) = |v|v$

`varFixed(value)`

value	~ vv
-------	------

`varFixedvarFunc`

<bates@stat.wisc.edu>

[varClassesvarWeights.varFuncvarFunc](#)

```
vf1 <- varFixed(~age)
```

varFunc

object[varFixed](#)object"varFunc"

varFunc(object)

object "varFunc"

"varFunc"

<bates@stat.wisc.edu>

[summary.varFunc](#)[varFixed](#)[varWeights](#).[varFunc](#)[coef](#).[varFunc](#)

vf1 <- varFunc(~age)

varIdent

varIdentformform $M > 1$ $M - 1$ value

varIdent(value, form, fixed)

value formvaluevaluenumeric(0)object
form ~ v~ v | gvgform*~ v | g1 * g2 * g3~ 1
fixed fixedNULL

varIdentvarFunc

<bates@stat.wisc.edu>

[varClasses](#)[varWeights](#).[varFunc](#)[coef](#).[varIdent](#)

vf1 <- varIdent(c(Female = 0.5), form = ~ 1 | Sex)

Variogram

"gls" "lme"

Variogram(object, distance, ...)

object

distance objectdistance(1,2), (1,3), ..., (n-1,n) nobjectn(n-1)/2

...

<bates@stat.wisc.edu>

Variogram.corExpVariogram.corGausVariogram.corLinVariogram.corRatio
Variogram.corSpatialVariogram.corSpherVariogram.defaultVariogram.gls
Variogram.lmeplot.Variogram

Variogram.corExp

objectdistance

S3 method for class 'corExp'

Variogram(object, distance, sig2, length.out, ...)

object "[corExp](#)"

distance NULLlength.outgetCovariate(object)

sig2 1

length.out distance = NULL50

...

variogdistVariogram

<bates@stat.wisc.edu>

[corExpplot.VariogramVariogram](#)

```
stopifnot(require("stats", quietly = TRUE))
cs1 <- corExp(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corGaus

objectdistance

```
## S3 method for class 'corGaus'
Variogram(object, distance, sig2, length.out, ...)
```

```
object      "corGaus"
distance     NULLlength.outgetCovariate(object)
sig2         1
length.out   distance = NULL50
...
```

variogdistVariogram

<bates@stat.wisc.edu>

[corGausplot.VariogramVariogram](#)

```
cs1 <- corGaus(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.corLin

objectdistance

```
## S3 method for class 'corLin'  
Variogram(object, distance, sig2, length.out, ...)
```

```
object      "corLin"  
distance     NULLlength.outgetCovariate(object)  
sig2         1  
length.out   distance = NULL50  
...
```

variogdistVariogram

<bates@stat.wisc.edu>

[corLinplot.VariogramVariogram](#)

```
cs1 <- corLin(15, form = ~ Time | Rat)  
cs1 <- Initialize(cs1, BodyWeight)  
Variogram(cs1)[1:10,]
```

Variogram.corRatio

objectdistance

```
## S3 method for class 'corRatio'  
Variogram(object, distance, sig2, length.out, ...)
```

```

object      "corRatio"
distance    NULLlength.outgetCovariate(object)
sig2        1
length.out  distance = NULL50
...

```

```

variogdistVariogram

```

```

<bates@stat.wisc.edu>

```

```

corRatioplot.VariogramVariogram

```

```

cs1 <- corRatio(7, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]

```

```

Variogram.corSpatial

```

```

FUNobjectdistance

```

```

## S3 method for class 'corSpatial'
Variogram(object, distance, sig2, length.out, FUN, ...)

```

```

object      "corSpatial"
distance    NULLlength.outgetCovariate(object)
sig2        1
length.out  distance = NULL50
FUN         object
...

```

```

variogdistVariogram

```

```

<bates@stat.wisc.edu>

```

[corSpatialVariogramVariogram.defaultVariogram.corExpVariogram.corGausVariogram.corLinVariogram.corRatioVariogram.corSpherplot.Variogram](#)

```
cs1 <- corExp(3, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1, FUN = function(x, y) (1 - exp(-x/y)))[1:10,]
```

Variogram.corSpher

objectdistance

```
## S3 method for class 'corSpher'
Variogram(object, distance, sig2, length.out, ...)
```

```
object          "corSpher"
distance         NULLlength.outgetCovariate(object)
sig2             1
length.out       distance = NULL50
...
```

variogdistVariogram

<bates@stat.wisc.edu>

[corSpherplot.VariogramVariogram](#)

```
cs1 <- corSpher(15, form = ~ Time | Rat)
cs1 <- Initialize(cs1, BodyWeight)
Variogram(cs1)[1:10,]
```

Variogram.default

objectdistance x, y object $(x - y)^2/2$

Default S3 method:
Variogram(object, distance, ...)

object
distance objectdistance(1,2), (1,3), ..., (n-1,n)nobjectn(n-1)/2
...

variogdistVariogram

<bates@stat.wisc.edu>

VariogramVariogram.glsVariogram.lmeplot.Variogram

```
fm1 <- lm(follicles ~ sin(2 * pi * Time) + cos(2 * pi * Time), Ovary,  
          subset = Mare == 1)  
Variogram(resid(fm1), dist(1:29))[1:10,]
```

Variogram.gls

glscollapse"none"robust = TRUE

S3 method for class 'gl'
Variogram(object, distance, form, resType, data,
 na.action, maxDist, length.out, collapse, nint, breaks,
 robust, metric, ...)

```

object      "gls"
distance    formdatametricobjectcorSpatialgetCovariate
form        |form~1
resType     "response""pearson""normalized""pearson"
data        formobject
na.action   NAna.fail
maxDist
length.out  objectcorSpatiallength.outVariogram50
collapse    "quantiles""fixed""none""quantiles"
nint        20
robust      TRUEFALSE
breaks      collapse
metric      "euclidean""maximum""manhattan""euclidean"
...

```

```

variogdistn.pairsobjectcorSpatial"modelVariog"Variogram

```

```

<bates@stat.wisc.edu>

```

```

glsVariogramVariogram.defaultVariogram.lmeplot.Variogram

```

```

fm1 <- gls(weight ~ Time * Diet, BodyWeight)
Vm1 <- Variogram(fm1, form = ~ Time | Rat)
print(head(Vm1), digits = 3)

```

```

Variogram.lme

```

```

lmecollapse"none"robust = TRUE

```

```

## S3 method for class 'lme'
Variogram(object, distance, form, resType, data,
          na.action, maxDist, length.out, collapse, nint, breaks,
          robust, metric, ...)

```

```

object      "lme"
distance    formdatametricobjectcorSpatialgetCovariate
form        |form~1
resType     "response""pearson""normalized""pearson"
data        formobject
na.action   NAna.fail
maxDist
length.out  objectcorSpatiallength.outVariogram50
collapse    "quantiles""fixed""none""quantiles"
nint        20
robust      TRUEFALSE
breaks      collapse
metric      "euclidean""maximum""manhattan""euclidean"
...

```

```

variogdistn.pairsobjectcorSpatial"modelVariog"Variogram

```

```

<bates@stat.wisc.edu>

```

```

lmeVariogramVariogram.defaultVariogram.glsplot.Variogram

```

```

fm1 <- lme(weight ~ Time * Diet, data=BodyWeight, ~ Time | Rat)
Variogram(fm1, form = ~ Time | Rat, nint = 10, robust = TRUE)

```

```

varPower

```

```

varPower  $v\sigma^2(v)v\sigma^2(v) = |v|^{2\theta}\theta\theta$ 

```

```

varPower(value, form, fixed)

```

```

value      Valueformvalueformformvaluenumeric(0)object
form       ~ v~ v | gvg"."fitted(.)resid(.)form*~ v | g1 * g2 * g3~ fitted(.)
fixed      formfixedfixedNULL

```

varPowervarFunc

<bates@stat.wisc.edu>

varWeights.varFunccoef.varPower

vf1 <- varPower(0.2, form = ~age|Sex)

varWeights

object

varWeights(object)

object varFunc

objectweightsNULL

<bates@stat.wisc.edu>

logLik.varFuncvarWeights

```
vf1 <- varPower(form=~age)
vf1 <- Initialize(vf1, Orthodont)
coef(vf1) <- 0.3
varWeights(vf1)[1:10]
```

varWeights.glsStruct

objectvarStructvarFunc

S3 method for class 'glsStruct'
varWeights(object)

object "glsStruct"corStruct"varFunc"

objectvarStruct

<bates@stat.wisc.edu>

varWeights

varWeights.lmeStruct

objectvarStructvarFunc

S3 method for class 'lmeStruct'
varWeights(object)

object "lmeStruct"reStructcorStructvarFunc

objectvarStruct

<bates@stat.wisc.edu>

varWeights

Wafer

Wafer

12345678910

12345678

Wheat

Wheat

312456897121110

Wheat2

Wheat2

4231

ARAPAHOE BRULEBUCKSKINCENTURACENTURK78CHEYENNECODYCOLTGAGEHOMESTEADKS831374
LANCERLANCOTANE83404NE83406NE83407NE83432NE83498NE83T12NE84557NE85556
NE85623NE86482NE86501NE86503NE86507NE86509NE86527NE86582NE86606NE86607
NE86T666NE87403NE87408NE87409NE87446NE87451NE87457NE87463NE87499NE87512
NE87513NE87522NE87612NE87613NE87615NE87619NE87627NORKANREDLANDROUGH RIDER
SCOUT66SIOUXLANDTAM107TAM200VONA

[.pdMat

x

```
## S3 method for class 'pdMat'  
x[i, j, drop = TRUE]  
## S3 replacement method for class 'pdMat'  
x[i, j] <- value
```

x	"pdMat"
ij	objectij
drop	TRUEFALSE
value	x

ijpdMatx

<bates@stat.wisc.edu>

[\[pdMat](#)

```
pd1 <- pdSymm(diag(3))  
pd1[1, , drop = FALSE]  
pd1[1:2, 1:2] <- 3 * diag(2)
```


nnet

class.ind

class.ind(cl)

cl

```
# The function is currently defined as
class.ind <- function(cl)
{
  n <- length(cl)
  cl <- as.factor(cl)
  x <- matrix(0, n, length(levels(cl)) )
  x[(1:n) + n*(unclass(cl)-1)] <- 1
  dimnames(x) <- list(names(cl), levels(cl))
  x
}
```

multinom

```
multinom(formula, data, weights, subset, na.action,  
          contrasts = NULL, Hess = FALSE, summ = 0, censored = FALSE,  
          model = FALSE, ...)
```

formula	response ~ predictors	formula()
data	formula	
weights		
subset		
na.action		
contrasts		
Hess		
summ	C	
censored	K	
model	model	
...	nnet	

[multinom](#)[nnet](#)

nnet	
deviance	
edf	
AIC	
Hessian	Hess
model	model

[nnet](#)

```
oc <- options(contrasts = c("contr.treatment", "contr.poly"))  
library(MASS)  
example(birthwt)  
(bwt.mu <- multinom(low ~ ., bwt))  
options(oc)
```

nnet

```
nnet(x, ...)  
  
## S3 method for class 'formula'  
nnet(formula, data, weights, ...,  
      subset, na.action, contrasts = NULL)  
  
## Default S3 method:  
nnet(x, y, weights, size, Wts, mask,  
      linout = FALSE, entropy = FALSE, softmax = FALSE,  
      censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,  
      maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,  
      abstol = 1.0e-4, reltol = 1.0e-8, ...)  
  
formula      class ~ x1 + x2 + ...  
x            x  
y  
weights  
size  
data         formula  
subset  
na.action    NA  
contrasts  
Wts  
mask  
linout  
entropy  
softmax      linoutentropysoftmaxcensored  
censored     softmaxsoftmax(0, 1, 1)censored  
skip  
rang         rangrangrang|x|  
decay  
maxit  
Hess         Hessian  
trace        TRUE  
MaxNWts      MaxNWts  
abstol       abstol  
reltol       1 - reltol  
...
```

formulannet.default

optim

"nnet" "nnet.formula"

wt

value

fitted.values

residuals

convergence 10

predict.nnetnnetHess

```
# use half the iris data
ir <- rbind(iris3[,1],iris3[,2],iris3[,3])
targets <- class.ind( c(rep("s", 50), rep("c", 50), rep("v", 50)) )
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
ir1 <- nnet(ir[samp,], targets[samp,], size = 2, rang = 0.1,
            decay = 5e-4, maxit = 200)
test.cl <- function(true, pred) {
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
test.cl(targets[-samp,], predict(ir1, ir[-samp,]))

# or
ird <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),
                  species = factor(c(rep("s",50), rep("c", 50), rep("v", 50))))
ir.nn2 <- nnet(species ~ ., data = ird, subset = samp, size = 2, rang = 0.1,
               decay = 5e-4, maxit = 200)
table(ird$species[-samp], predict(ir.nn2, ird[-samp,], type = "class"))
```

nnetHess

Hess=TRUEnnetvcov.multinom

nnetHess(net, x, y, weights)

net nnetnnet

x

y

weights nnet

[nnetpredict.nnet](#)

```
# use half the iris data
ir <- rbind(iris3[,1], iris3[,2], iris3[,3])
targets <- matrix(c(rep(c(1,0,0),50), rep(c(0,1,0),50), rep(c(0,0,1),50)),
  150, 3, byrow=TRUE)
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
ir1 <- nnet(ir[samp,], targets[samp,], size=2, rang=0.1, decay=5e-4, maxit=200)
eigen(nnetHess(ir1, ir[samp,], targets[samp,]), TRUE)$values
```

predict.nnet

```
## S3 method for class 'nnet'
predict(object, newdata, type = c("raw","class"), ...)
```

object nnetnnet

newdata

type

...


```
predict()"nnet"predict(x)xpredict.nnet(x)
```

```
type = "raw"type = "class"nnet.formula
```

```
nnetwhich.is.max
```

```
# use half the iris data
ir <- rbind(iris3[,1], iris3[,2], iris3[,3])
targets <- class.ind( c(rep("s", 50), rep("c", 50), rep("v", 50)) )
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
ir1 <- nnet(ir[samp,], targets[samp,],size = 2, rang = 0.1,
            decay = 5e-4, maxit = 200)
test.cl <- function(true, pred){
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
test.cl(targets[-samp,], predict(ir1, ir[-samp,]))

# or
ird <- data.frame(rbind(iris3[,1], iris3[,2], iris3[,3]),
                  species = factor(c(rep("s",50), rep("c", 50), rep("v", 50))))
ir.nn2 <- nnet(species ~ ., data = ird, subset = samp, size = 2, rang = 0.1,
               decay = 5e-4, maxit = 200)
table(ird$species[-samp], predict(ir.nn2, ird[-samp,], type = "class"))
```

```
which.is.max
```

```
which.is.max(x)
```

```
x
```

max.colwhich.max

```
## Not run: ## this is incomplete
pred <- predict(nnet, test)
table(true, apply(pred, 1, which.is.max))

## End(Not run)
```


rpart

```
car.test.frame
```

```
car.test.framecu.summary
```

```
car.test.frame
```

```
Price
Country FranceGermanyJapanJapan/USAKoreaMexicoSwedenUSA
Reliability 15
Mileage
Type CompactLargeMediumSmallSportyVan
Weight
Disp.
HP
```

```
car90cu.summary
```

```
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
summary(z.auto)
```

car90

data(car90)

Much worseworseaveragebetterMuch betterNA

man.4man.5man.6
auto.3auto.4auto.CVT

SmallSportyCompactMediumLargeVan

car.all

[car.test.framecu.summary](#)

```
data(car90)
plot(car90$Price/1000, car90$Weight,
     xlab = "Price (thousands)", ylab = "Weight (lbs)")
mlowess <- function(x, y, ...) {
  keep <- !(is.na(x) | is.na(y))
  lowess(x[keep], y[keep], ...)
}
with(car90, lines(mlowess(Price/1000, Weight, f = 0.5)))
```

cu.summary

cu.summary

cu.summary

Price

Country BrazilEnglandFranceGermanyJapanJapan/USAKoreaMexicoSwedenUSA

Reliability Much worseworseaveragebetterMuch better

Mileage

Type CompactLargeMediumSmallSportyVan

[car.test.framecar90](#)

```
fit <- rpart(Price ~ Mileage + Type + Country, cu.summary)
par(xpd = TRUE)
plot(fit, compress = TRUE)
text(fit, use.n = TRUE)
```

kyphosis

kyphosis

kyphosis

Kyphosis absentpresent

Age

Number

Start

```
fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
fit2 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
  parms = list(prior = c(0.65, 0.35), split = "information"))
fit3 <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
  control = rpart.control(cp = 0.05))
par(mfrow = c(1,2), xpd = TRUE)
plot(fit)
text(fit, use.n = TRUE)
plot(fit2)
text(fit2, use.n = TRUE)
```

labels.rpart

rpart

```
## S3 method for class 'rpart'
labels(object, digits = 4, minlength = 1L, pretty, collapse = TRUE, ...)
```

object	"rpart"rpart
digits	rpartoptions("digits")
minlength	01abbreviateminlength
pretty	pretty = 0minlength = 0Lpretty = NULLminlength = 1Lpretty = TRUE minlength = 4L
collapse	collapse = TRUE"root" FALSE"leaf"
...	abbreviate

```
collapse = TRUEcollapse = FALSErpartrpart
```

[abbreviate](#)

`meanvar.rpart`

```
meanvar(tree, ...)
```

```
## S3 method for class 'rpart'
```

```
meanvar(tree, xlab = "ave(y)", ylab = "ave(deviance)", ...)
```

```
tree          "rpart"rpart
```

```
xlab
```

```
ylab
```

```
...
```

```
x              yval
```

```
y
```

```
label
```

[plot.rpart](#)

```
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
```

```
meanvar(z.auto, log = 'xy')
```

`na.rpart`

`"rpart"``na.rpart(x)``x``rpart`

`path.rpart`

`path.rpart(tree, nodes, pretty = 0, print.it = TRUE)`

<code>tree</code>	<code>"rpart"</code>	<code>rpart</code>
-------------------	----------------------	--------------------

<code>nodes</code>		
--------------------	--	--

<code>pretty</code>	<code>NULL</code>	
---------------------	-------------------	--

<code>print.it</code>	<code>nodes</code>	
-----------------------	--------------------	--

`rpart``rpart``path.tree``rpart`

```
fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
print(fit)
path.rpart(fit, nodes = c(11, 22))
```

plot.rpart

```
## S3 method for class 'rpart'
plot(x, uniform = FALSE, branch = 1, compress = FALSE, nspace,
      margin = 0, minbranch = 0.3, branch.col = 1, branch.lty = 1,
      branch.lwd = 1, ...)
```

x	"rpart"
uniform	TRUE
branch	
compress	FALSE1:nleavesTRUEuniform=TRUE
nspace	branch
margin	
minbranch	minbranchuniform=TRUE
branch.col	
branch.lty	
branch.lwd	
...	

plotrpart

xy

text

[rparttext.rpart](#)

```
fit <- rpart(Price ~ Mileage + Type + Country, cu.summary)
par(xpd = TRUE)
plot(fit, compress = TRUE)
text(fit, use.n = TRUE)
```

plotcp

rpart

```
plotcp(x, minline = TRUE, lty = 3, col = 1,  
       upper = c("size", "splits", "none"), ...)
```

x "rpart"

minline

lty

col

upper

...

[cprpartcptablecp](#)

[rpartprintcprpart.object](#)

post.rpart

rpart

```
post(tree, ...)
```

```
## S3 method for class 'rpart'
```

```
post(tree, title.,  
      filename = paste(deparse(substitute(tree)), ".ps", sep = ""),  
      digits = getOption("digits") - 2, pretty = TRUE,  
      use.n = TRUE, horizontal = TRUE, ...)
```

```

tree          "rpart"rpart
title.        rpart
filename      rpart.psfilename = ""
digits
pretty        NULLTRUE
use.n         TRUE#events level1/ #events level2/classnanova#events/npoisson
exp
horizontal    TRUEFALSE
...           postscript

```

```

plot.rparttext.rpartfancyrpart

```

```

rpartpostscriptfilename = ""

```

[plot.rpartrparttext.rpartabbreviate](#)

```

## Not run:
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
post(z.auto, file = "") # display tree on active device
# now construct postscript version on file "pretty.ps"
# with no title
post(z.auto, file = "pretty.ps", title = " ")
z.hp <- rpart(Mileage ~ Weight + HP, car.test.frame)
post(z.hp)
## End(Not run)

```

```

predict.rpart

```

```

rpart

```

```

## S3 method for class 'rpart'
predict(object, newdata,
        type = c("vector", "prob", "class", "matrix"),
        na.action = na.pass, ...)

```

```

object      "rpart"rpart
newdata     formula(object)newdata
type        rpartprobttree
na.action   newdatana.omitna.fail
...

```

```
"rpart"predictpredict.rpart
```

```
newdataframe$yval
```

```
type = "vector"
```

```
type = "prob"
```

```
type = "matrix"
```

```
frame$yval2frame$yval
```

```
type = "class"
```

[predictrpart.object](#)

```
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
```

```
predict(z.auto)
```

```
fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
```

```
predict(fit, type = "prob") # class probabilities (default)
```

```
predict(fit, type = "vector") # level numbers
```

```
predict(fit, type = "class") # factor
```

```
predict(fit, type = "matrix") # level number, class frequencies, probabilities
```

```
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
```

```
fit <- rpart(Species ~ ., data = iris, subset = sub)
```

```
fit
```

```
table(predict(fit, iris[-sub,], type = "class"), iris[-sub, "Species"])
```

```
print.rpart
```

```
rpartprint"rpart"
```

```
## S3 method for class 'rpart'
```

```
print(x, minlength = 0, spaces = 2, cp, digits = getOption("digits"),
```

```
      nsmall = min(20, digits), ...)
```

```
x                "rpart"rpart
```

```
minlength        labels.rpart
```

```
spaces
```

```
digits
```

```
nsmall           format
```

```
cp               cp
```

```
...
```

```
print"rpart"print.rpart
```

```
x$frame"class"
```

```
printrpart.objectsummary.rpartprintcp
```

```
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
z.auto
## Not run: node), split, n, deviance, yval
      * denotes terminal node

1) root 60 1354.58300 24.58333
  2) Weight>=2567.5 45  361.20000 22.46667
    4) Weight>=3087.5 22   61.31818 20.40909 *
    5) Weight<3087.5 23  117.65220 24.43478
      10) Weight>=2747.5 15   60.40000 23.80000 *
      11) Weight<2747.5 8   39.87500 25.62500 *
    3) Weight<2567.5 15  186.93330 30.93333 *
```

```
## End(Not run)
```

```
printcp
```

```
cprpart
```

```
printcp(x, digits = getOption("digits") - 2)
```

```
x          "rpart"rpart
digits
```

```
summary.rpartrpart.object
```

```

z.auto <- rpart(Mileage ~ Weight, car.test.frame)
printcp(z.auto)
## Not run:
Regression tree:
rpart(formula = Mileage ~ Weight, data = car.test.frame)

Variables actually used in tree construction:
[1] Weight

Root node error: 1354.6/60 = 22.576

      CP nsplit rel error  xerror   xstd
1 0.595349      0  1.00000 1.03436 0.178526
2 0.134528      1  0.40465 0.60508 0.105217
3 0.012828      2  0.27012 0.45153 0.083330
4 0.010000      3  0.25729 0.44826 0.076998

## End(Not run)

```

prune.rpart

rpartsnippingcp

prune(tree, ...)

```

## S3 method for class 'rpart'
prune(tree, cp, ...)

```

```

tree      "rpart"rpart
cp        rpart
...

```

rpartcp

[rpart](#)

```

z.auto <- rpart(Mileage ~ Weight, car.test.frame)
zp <- prune(z.auto, cp = 0.1)
plot(zp) #plot smaller rpart object

```

`residuals.rpart`

`residualsrpart`

```
## S3 method for class 'rpart'
residuals(object, type = c("usual", "pearson", "deviance"), ...)
```

```
object      "rpart"
type
            anovay - fitteduser
            usualpearsondeviance
            poissonexpusualpearsondeviance
...
```

`typerpart`

```
fit <- rpart(skips ~ Opening + Solder + Mask + PadType + Panel,
             data = solder.balance, method = "anova")
summary(residuals(fit))
plot(predict(fit),residuals(fit))
```

`rpart`

`rpart`

```
rpart(formula, data, weights, subset, na.action = na.rpart, method,
      model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```



```

formula      model.frame).
data
weights
subset
na.action    y
method       "anova""poisson""class""exp"methodmethod = "exp"ymethod =
             "poisson"ymethod = "class"method = "anova"
             methodinitsplitevaltests/usersplits.R
model        modelrpart
x            x
y            modelFALSE
parms

```

```

             priorlosssplitginiinformationgini
control      rpartrpart.control
cost
...          rpart.controlrpart

```

```

tree
tree

```

```

rpartrpart.object

```

```

rpart.controlrpart.objectssummary.rpartprint.rpart

```

```

fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
fit2 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
             parms = list(prior = c(.65,.35), split = "information"))
fit3 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
             control = rpart.control(cp = 0.05))
par(mfrow = c(1,2), xpd = NA) # otherwise on some devices the text is clipped
plot(fit)
text(fit, use.n = TRUE)
plot(fit2)
text(fit2, use.n = TRUE)

```

`rpart.control`

`rpart`

```
rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01,  
              maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10,  
              surrogatestyle = 0, maxdepth = 30, ...)
```

```
minsplit  
minbucket      <leaf>minbucketminsplitminsplitminbucket*3minbucketminsplit/3  
cp             cpanovacpp  
maxcompete  
maxsurrogate  
usesurrogate   0120tree2  
xval  
surrogatestyle 01  
maxdepth  
...
```

[rpart](#)

`rpart.exp`

```
rpart.exp(y, offset, parms, wt)
```

```
y           Surv  
offset  
parms       shrinkmethod"deviance""sqrt""deviance"  
wt
```

[rpart](#)

rpart.object

rpart

frame row.namesframeframevar"<leaf>"nwtdevyvalsplitscomplexity
 ncompetensurrogate
 yval2
where frame
call update(tree)
terms c("terms", "formula")[terms.object](#)
splits countncat+/-1improveindexadjindexncatx < cutpointx > cutpoint
csplit indexsplits132
method "class""exp""poisson""anova""user"
cptable
variable.importance
 [summary.rpart](#)
numresp
parmscontrol
functions summaryprinttext
ordered
na.action [model.frame](#)NA na.action
"xlevels""levels"
modelxyrpart

rpart

[rpart](#)

rsq.rpart

rsq.rpart(x)

x "rpart"rpart

"anova"

```
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
rsq.rpart(z.auto)
```

snip.rpart

snip.rpart(x, toss)

x "rpart"rpart
toss

rpartrpart

rpart

[plot.rpart](#)

```
## dataset not in R
## Not run:
z.survey <- rpart(market.survey) # grow the rpart object
plot(z.survey) # plot the tree
z.survey2 <- snip.rpart(z.survey, toss = 2) # trim subtree at node 2
plot(z.survey2) # plot new tree

# can also interactively select the node using the mouse in the
# graphics window

## End(Not run)
```

solder.balance

solder.balance

solder

solder

Opening LMS

Solder ThickThin

Mask A1.5A3B3B6

PadType D4D6D7L4L6L7L8L9W4W9

Panel 1:3

skips

```
fit <- rpart(skips ~ Opening + Solder + Mask + PadType + Panel,
             data = solder.balance, method = "anova")
summary(residuals(fit))
plot(predict(fit), residuals(fit))
```

stagec

data(stagec)

pgtime

pgstat

age

eet

g2

grade

gleason

ploidy diploidtetraploidaneuploid

```
require(survival)
rpart(Surv(pgtime, pgstat) ~ ., stagec)
```

summary.rpart

rpart

```
## S3 method for class 'rpart'
summary(object, cp = 0, digits = getOption("digits"), file, ...)
```

```
object          "rpart"rpart
digits
cp              cp
file
...
```

```
"rpart"summarysummary.rpart
```

```
printcp
```

```
summaryrpart.objectprintcp
```

```
## a regression tree
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
summary(z.auto)
```

```
## a classification tree with multiple variables and surrogate splits.
summary(rpart(Kyphosis ~ Age + Number + Start, data = kyphosis))
```

text.rpart

```
## S3 method for class 'rpart'
text(x, splits = TRUE, label, FUN = text, all = FALSE,
     pretty = NULL, digits = getOption("digits") - 3, use.n = FALSE,
     fancy = FALSE, fwidth = 0.8, fheight = 0.8, bg = par("bg"),
     minlength = 1L, ...)
```

x	"rpart"rpart
splits	TRUE
label	rpart2
FUN	text
all	TRUE
minlength	labels.rpart
pretty	minlength labels.rpart
digits	
use.n	TRUE(#events level1/ #events level2/classnanova#events/npoisson exp
fancy	TRUEyval
fwidth	fancyfwidth < 1fwidth > 1
fheight	fancyfheight < 1fheight > 1
bg	fancy = TRUE
...	parxpd = TRUE

[textplot.rpart](#)[rpart](#)[labels.rpart](#)[abbreviate](#)

```
freen.tr <- rpart(y ~ ., freeny)
par(xpd = TRUE)
plot(freen.tr)
text(freen.tr, use.n = TRUE, all = TRUE)
```

`xpred.rpart`

`rpart``xpred.rpart(fit, xval = 10, cp, return.all = FALSE)``fit` `"rpart"``xval``cp` `cptable``return.all``cp.36.28.13[.36,1]cp[.28,.36][.13,.28)``return.allresult[1,,]result[2,,]``rpart`

```
fit <- rpart(Mileage ~ Weight, car.test.frame)
xmat <- xpred.rpart(fit)
xerr <- (xmat - car.test.frame$Mileage)^2
apply(xerr, 2, sum)  # cross-validated error estimate

# approx same result as rel. error from printcp(fit)
apply(xerr, 2, sum)/var(car.test.frame$Mileage)
printcp(fit)
```


spatial

```
anova.trls
```

```
anova.trlsanovaalist.trls
```

```
## S3 method for class 'trls'  
anova(object, ...)  
anovaalist.trls(object, ...)
```

```
object      surf.ls  
...
```

```
anova.trlsanovaalist.trls
```

```
surf.ls
```

```
library(stats)  
data(topo, package="MASS")  
topo0 <- surf.ls(0, topo)  
topo1 <- surf.ls(1, topo)  
topo2 <- surf.ls(2, topo)  
topo3 <- surf.ls(3, topo)  
topo4 <- surf.ls(4, topo)  
anova(topo0, topo1, topo2, topo3, topo4)  
summary(topo4)
```

correlogram

```
correlogram(krig, nint, plotit = TRUE, ...)
```

```
krig          xyz
```

```
nint
```

```
plotit
```

```
...
```

```
nint
```

```
xycnt
```

```
plotit = TRUE
```

variogram

```
data(topo, package="MASS")
topo.kr <- surf.ls(2, topo)
correlogram(topo.kr, 25)
d <- seq(0, 7, 0.1)
lines(d, expcov(d, 0.7))
```

expcov

surf.gls

```
expcov(r, d, alpha = 0, se = 1)
gaucov(r, d, alpha = 0, se = 1)
sphercov(r, d, alpha = 0, se = 1, D = 2)
```

r
d
alpha
se
D

surf.gls

```
data(topo, package="MASS")
topo.kr <- surf.ls(2, topo)
correlogram(topo.kr, 25)
d <- seq(0, 7, 0.1)
lines(d, expcov(d, 0.7))
```

Kaver

Kaver(fs, nsim, ...)

fs
nsim
...

xy

KfnKenvl

```
towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 40), type="b")
plot(Kfn(towns, 10), type="b", xlab="distance", ylab="L(t)")
for(i in 1:10) lines(Kfn(Psim(69), 10))
lims <- Kenvl(10,100,Psim(69))
lines(lims$x,lims$lower, lty=2, col="green")
lines(lims$x,lims$upper, lty=2, col="green")
lines(Kaver(10,25,Strauss(69,0.5,3.5)), col="red")
```

Kenvl

Kenvl(fs, nsim, ...)

fs
nsim
...

x
lower
upper
aver

KfnKaver

```

towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 40), type="b")
plot(Kfn(towns, 10), type="b", xlab="distance", ylab="L(t)")
for(i in 1:10) lines(Kfn(Psim(69), 10))
lims <- Kenvl(10,100,Psim(69))
lines(lims$x,lims$lower, lty=2, col="green")
lines(lims$x,lims$upper, lty=2, col="green")
lines(Kaver(10,25,Strauss(69,0.5,3.5)), col="red")

```

Kfn

$$L = \sqrt{K/\pi}$$

Kfn(pp, fs, k=100)

pp	xy
fs	
k	fs

ppinitppregion

x	
y	
k	kfs
dmin	
lm	

ppinitppregionKaverKenvl

```

towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 10), type="s", xlab="distance", ylab="L(t)")

```

ppgetregion

$(x_l, x_u) \times (y_l, y_u)$ C

ppgetregion()

c("xl", "xu", "yl", "yu")

ppregion

ppinit

ppinit(file)

file

"pp"xyxlxuylyu

ppregion

ppregion

```
towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 10), type="b", xlab="distance", ylab="L(t)")
```

pplik

```
pplik(pp, R, ng=50, trace=FALSE)
```

```
pp
R          R
ng          ngngR
trace
```

```
c[0,1]
```

[Strauss](#)

```
pines <- ppinit("pines.dat")
pplik(pines, 0.7)
```

ppregion

```
(xl, xu) × (yl, yu)
```

```
ppregion(xl = 0, xu = 1, yl = 0, yu = 1)
```

```
xl          xlxlyu
xuylyu
```

C

[ppinit](#)[ppgetregion](#)

predict.trls

```
## S3 method for class 'trls'
predict(object, x, y, ...)
```

```
object      surf.ls
x
y
...
```

predict.trlsimagecontourtrmat

[surf.lstrmat](#)

```
data(topo, package="MASS")
topo2 <- surf.ls(2, topo)
topo4 <- surf.ls(4, topo)
x <- c(1.78, 2.21)
y <- c(6.15, 6.15)
z2 <- predict(topo2, x, y)
z4 <- predict(topo4, x, y)
cat("2nd order predictions:", z2, "\n4th order predictions:", z4, "\n")
```

prmat

```
prmat(obj, xl, xu, yl, yu, n)
```

```
obj      surf.gls
xl
xu
yl
yu
n        nn
```

xyzcontourimage

[surf.glstrmatsemat](#)

```
data(topo, package="MASS")
topo.kr <- surf.gls(2, expcov, topo, d=0.7)
prsurf <- prmat(topo.kr, 0, 6.5, 0, 6.5, 50)
contour(prsurf, levels=seq(700, 925, 25))
```

Psim

Psim(n)

n

ppinitppregion

xy

[SSIStrauss](#)

```
towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 10), type="s", xlab="distance", ylab="L(t)")
for(i in 1:10) lines(Kfn(Psim(69), 10))
```

semat

```
semat(obj, xl, xu, yl, yu, n, se)
```

```
obj          surf.gls
xl
xu
yl
yu
n            nn
se
```

```
contourimage
```

[surf.glstrmatprmat](#)

```
data(topo, package="MASS")
topo.kr <- surf.gls(2, expcov, topo, d=0.7)
prsurf <- prmat(topo.kr, 0, 6.5, 0, 6.5, 50)
contour(prsurf, levels=seq(700, 925, 25))
sesurf <- semat(topo.kr, 0, 6.5, 0, 6.5, 30)
contour(sesurf, levels=c(22,25))
```

SSI

```
SSI(n, r)
```

```
n
r
```

ppinitppregion

xy

rn

PsimStrauss

```
towns <- ppinit("towns.dat")
par(pty = "s")
plot(Kfn(towns, 10), type = "b", xlab = "distance", ylab = "L(t)")
lines(Kaver(10, 25, SSI(69, 1.2)))
```

Strauss

Strauss(n , $c=0$, r)

n

c $c[0,1]c = 0r$

r

nnppinitppregion

xy

PsimSSI

```
towns <- ppinit("towns.dat")
par(pty="s")
plot(Kfn(towns, 10), type="b", xlab="distance", ylab="L(t)")
lines(Kaver(10, 25, Strauss(69,0.5,3.5)))
```

surf.gls

```
surf.gls(np, covmod, x, y, z, nx = 1000, ...)
```

```
np
covmod
x      xyz
y
z      x$z
nx
...    covmod
```

```
beta
x
y
z
```

[trmat](#)[surf.ls](#)[prmat](#)[sema](#)[texpcov](#)[gaucov](#)[spher](#)[cov](#)

```
library(MASS) # for eqscplot
data(topo, package="MASS")
topo.kr <- surf.gls(2, expcov, topo, d=0.7)
trsurf <- trmat(topo.kr, 0, 6.5, 0, 6.5, 50)
eqscplot(trsurf, type = "n")
contour(trsurf, add = TRUE)

prsurf <- prmat(topo.kr, 0, 6.5, 0, 6.5, 50)
contour(prsurf, levels=seq(700, 925, 25))
sesurf <- semat(topo.kr, 0, 6.5, 0, 6.5, 30)
eqscplot(sesurf, type = "n")
contour(sesurf, levels = c(22, 25), add = TRUE)
```

surf.ls

surf.ls(np, x, y, z)

np

x xyz

y

z x\$z

beta

x

y

z

[trmatsurf.gls](#)

```

library(MASS) # for eqscplot
data(topo, package="MASS")
topo.kr <- surf.ls(2, topo)
trsurf <- trmat(topo.kr, 0, 6.5, 0, 6.5, 50)
eqscplot(trsurf, type = "n")
contour(trsurf, add = TRUE)
points(topo)

eqscplot(trsurf, type = "n")
contour(trsurf, add = TRUE)
plot(topo.kr, add = TRUE)
title(xlab= "Circle radius proportional to Cook's influence statistic")

```

trls.influence

surf.ls

```

trls.influence(object)
## S3 method for class 'trls'
plot(x, border = "red", col = NA, pch = 4, cex = 0.6,
      add = FALSE, div = 8, ...)

```

```

objectx      surf.ls
div           plot.trls
add           TRUE
bordercolpchcex...

```

trls.influence

```

r             residuals.trls
hii
stresid
Di

```

[surf.lsinfluence.measuresplot.lm](#)

```

library(MASS) # for eqscplot
data(topo, package = "MASS")
topo2 <- surf.ls(2, topo)
infl.topo2 <- trls.influence(topo2)
(cand <- as.data.frame(infl.topo2)[abs(infl.topo2$stresid) > 1.5, ])
cand.xy <- topo[as.integer(rownames(cand)), c("x", "y")]
trsurf <- trmat(topo2, 0, 6.5, 0, 6.5, 50)
eqscplot(trsurf, type = "n")
contour(trsurf, add = TRUE, col = "grey")
plot(topo2, add = TRUE, div = 3)
points(cand.xy, pch = 16, col = "orange")
text(cand.xy, labels = rownames(cand.xy), pos = 4, offset = 0.5)

```

trmat

```
trmat(obj, xl, xu, yl, yu, n)
```

```

obj          surf.lssurf.gls
xl
xu
yl
yu
n            nn

```

xyzcontourimage

[surf.lssurf.gls](#)

```

data(topo, package="MASS")
topo.kr <- surf.ls(2, topo)
trsurf <- trmat(topo.kr, 0, 6.5, 0, 6.5, 50)

```

`variogram`

```
variogram(krig, nint, plotit = TRUE, ...)
```

```
krig          xyz  
nint  
plotit  
...
```

```
nint
```

```
xycnt
```

```
plotit = TRUE
```

[correlogram](#)

```
data(topo, package="MASS")  
topo.kr <- surf.ls(2, topo)  
variogram(topo.kr, 25)
```

survival

aareg	
"aareg"	
<pre>aareg(formula, data, weights, subset, na.action, qrtol=1e-07, nmin, dfbeta=FALSE, taper=1, test = c('aalen', 'variance', 'nrisk'), cluster, model=FALSE, x=FALSE, y=FALSE)</pre>	
formula	+Surv
data	formulasubsetweightsdata
weights	weightssubset
subset	
na.action	model.fr amesubsetna.failna.excl ude
qrtol	
nmin	
dfbeta	cluster
taper	taper=(1:4)/4
test	
cluster	
modelxy	
taper	

"aareg"

n

times

nrisk times

coefficient

test.statistic

test.var

test test

tweight

call

Fit a model to the lung cancer data set

```
lfit <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
              nmin=1)
```

Not run:

lfit

Call:

```
aareg(formula = Surv(time, status) ~ age + sex + ph.ecog, data = lung, nmin = 1
      )
```

n=227 (1 observations deleted due to missing values)

138 out of 138 unique event times used

	slope	coef	se(coef)	z	p
Intercept	5.26e-03	5.99e-03	4.74e-03	1.26	0.207000
age	4.26e-05	7.02e-05	7.23e-05	0.97	0.332000
sex	-3.29e-03	-4.02e-03	1.22e-03	-3.30	0.000976
ph.ecog	3.14e-03	3.80e-03	1.03e-03	3.70	0.000214

Chisq=26.73 on 3 df, p=6.7e-06; test weights=aalen

```
plot(lfit[4], ylim=c(-4,4)) # Draw a plot of the function for ph.ecog
```

End(Not run)

```
lfit2 <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
              nmin=1, taper=1:10)
```

Not run: lines(lfit2[4], col=2) # Nearly the same, until the last point

A fit to the multiple-infection data set of children with

Chronic Granulomatous Disease. See section 8.5 of Therneau and Grambsch.

```
fita2 <- aareg(Surv(tstart, tstop, status) ~ treat + age + inherit +
```

```

steroids + cluster(id), data=cgd)

## Not run:
n= 203
69 out of 70 unique event times used

      slope      coef se(coef) robust se      z      p
Intercept    0.004670  0.017800 0.002780  0.003910  4.55 5.30e-06
treatrIFN-g  -0.002520 -0.010100 0.002290  0.003020 -3.36 7.87e-04
age          -0.000101 -0.000317 0.000115  0.000117 -2.70 6.84e-03
inheritautosomal 0.001330  0.003830 0.002800  0.002420  1.58 1.14e-01
steroids      0.004620  0.013200 0.010600  0.009700  1.36 1.73e-01

Chisq=16.74 on 4 df, p=0.0022; test weights=aalen

## End(Not run)

```

aeqSurv

```
aeqSurv(x, tolerance = sqrt(.Machine$double.eps))
```

```
x
```

```
tolerance
```

```
survfitcoxph.timefix.survfitcoxph.control
```

```
all.equal
```

```
survfitcoxph.control
```

aggregate.survfit

```
## S3 method for class 'survfit'
aggregate(x, by = NULL, FUN = mean, ...)
```

```
x          survfit
by          dim(x)['data']
FUN
...
```

survfit

survfit

```
cfit <- coxph(Surv(futime, death) ~ sex + age*hgb, data=mgus2)
# marginal effect of sex, after adjusting for the others
dummy <- rbind(mgus2, mgus2)
dummy$sex <- rep(c("F", "M"), each=nrow(mgus2)) # population data set
dummy <- na.omit(dummy) # don't count missing hgb in our "population"
csurv <- survfit(cfit, newdata=dummy)
dim(csurv) # 2 * 1384 survival curves
csurv2 <- aggregate(csurv, dummy$sex)
```

agreg.fit

coxph

```
agreg.fit(x, y, strata, offset, init, control, weights, method,
rownames, resid=TRUE, nocenter=NULL)
coxph.fit(x, y, strata, offset, init, control, weights, method,
rownames, resid=TRUE, nocenter=NULL)
```

x
y Surv
strata
offset
init
control coxph.control
weights
method
rownames
resid
nocenter

residconcordance

coxph

aml

aml
leukemia
data(cancer, package="survival")

`anova.coxph`

```
## S3 method for class 'coxph'  
anova(object, ..., test = 'Chisq')
```

```
object      coxph  
...         coxph  
test
```

`anova`

"anova" "data.frame"

`anova`

[coxphanova](#)

```
fit <- coxph(Surv(futime, fustat) ~ resid.ds *rx + ecog.ps, data = ovarian)  
anova(fit)  
fit2 <- coxph(Surv(futime, fustat) ~ resid.ds +rx + ecog.ps, data=ovarian)  
anova(fit2,fit)
```

`attrassign`

"assign"

```
attrassign(object, ...)  
## Default S3 method:  
attrassign(object, tt,...)  
## S3 method for class 'lm'  
attrassign(object,...)
```

```
object  
tt  
...
```

```
survreg(Surv(time, status) ~ age + sex + factor(ph.ecog), lung)
```

```
$(Intercept)      1  
$age              2  
$sex              3  
$factor(ph.ecog) 4 5 6
```

[termsmodel.matrix](#)

```
formula <- Surv(time,status)~factor(ph.ecog)  
tt <- terms(formula)  
mf <- model.frame(tt,data=lung)  
mm <- model.matrix(tt,mf)  
## a few rows of data  
mm[1:3,]  
## old-style assign attribute  
attr(mm,"assign")  
## alternate style assign attribute  
attrassign(mm,tt)
```

basehaz

```
basehaz(fit, newdata, centered=TRUE)
```

```
fit  
newdata  
centered      newdatafit$mean
```



```
survfit.coxph  
timecumhazsurvfit  
exp(sum(coef(fit) * (x-z)))  
centered= FALSE
```

```
hazardtimestrata
```

```
survfit.coxph
```

```
bladder
```

```
bladder1  
bladder  
bladder2  
data(cancer, package="survival")
```

blogit

```
blogit(edge = 0.05)
bprobit(edge= 0.05)
bcloglog(edge=.05)
blog(edge=.05)
```

```
edge          blog
```

```
glmfamily = gaussian(link= "logit")familylinkfun
etastartmustartblogit
```

```
familymake.family
```

```
stats{make.family}
```

```
py <- pseudo(survfit(Surv(time, status) ~1, lung), time=730) #2 year survival
range(py)
pfit <- glm(py ~ ph.ecog, data=lung, family=gaussian(link=blogit()))
# For each +1 change in performance score, the odds of 2 year survival
# are multiplied by 1/2 = exp of the coefficient.
```

brier

```
brier(fit, times, newdata, ties = TRUE, detail = FALSE, timefix = TRUE,
      efron = FALSE)
```

```
fit          coxph
times
newdata
ties
detail
timefix
efron
```

```

$$R^2 = 1 - \sum (y - \hat{y})^2 / \sum (y - \mu)^2$$

tied
coxphefron
```

```
rsquared       $R^2$ 
brier
times
```

rttright

```
cfit <- coxph(Surv(rtime, recur) ~ age + meno + size + pmin(nodes,11),
              data= rotterdam)
round(cfit$concordance["concordance"], 3) # some predictive power
brier(cfit, times=c(4,6)*365.25) # values at 4 and 6 years
```

cch

```
cch(formula, data, subcoh, id, stratum=NULL, cohort.size,
    method =c("Prentice","SelfPrentice","LinYing","I.Borgan","II.Borgan"),
    robust=FALSE)
```

```
formula      Surv"right""counting"
subcoh        1TRUE0FALSEdatasubcoh
id
stratum
cohort.size
data
method
robust        "LinYing"robust=TRUE
```

data

```
coef
naive.var
var
```

O

twophasesvycoxph<http://faculty.washington.edu/tlumley/survey/>

```

## The complete Wilms Tumor Data
## (Breslow and Chatterjee, Applied Statistics, 1999)
## subcohort selected by simple random sampling.
##

subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel==1|subcoh==1)
ccoh.data <- nwtco[selccoh,]
ccoh.data$subcohort <- subcoh[selccoh]
## central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol, labels=c("FH", "UH"))
## tumour stage
ccoh.data$stage <- factor(ccoh.data$stage, labels=c("I", "II", "III", "IV"))
ccoh.data$age <- ccoh.data$age/12 # Age in years

##
## Standard case-cohort analysis: simple random subcohort
##

fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
  subcoh = ~subcohort, id=~seqno, cohort.size=4028)

fit.ccP

fit.ccSP <- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
  subcoh = ~subcohort, id=~seqno, cohort.size=4028, method="SelfPren")

summary(fit.ccSP)

##
## (post-)stratified on instit
##
stratsizes<-table(nwtco$instit)
fit.BI<- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
  subcoh = ~subcohort, id=~seqno, stratum=~instit, cohort.size=stratsizes,
  method="I.Borgan")

summary(fit.BI)

```

cgd

cgd
data(cgd)

cgd0cgd

link{cgd0}

cgd0

cgd0

cgdraw
cgd

cgd

cipoisson

```
cipoisson(k, time = 1, p = 0.95, method = c("exact", "anscombe"))
```

k
time
p
method

k

ktimekk

ppoisqpois

```
cipoisson(4) # 95% confidence limit
# lower    upper
# 1.089865 10.24153
ppois(4, 10.24153)    #chance of seeing 4 or fewer events with large rate
# [1] 0.02500096
1-ppois(3, 1.08986)   #chance of seeing 4 or more, with a small rate
# [1] 0.02499961
```

clogit

case.status~exposure+strata(matched.set)

```
clogit(formula, data, weights, subset, na.action,  
       method=c("exact", "approximate", "efron", "breslow"),  
       ...)
```

formula

data

weights

subset

na.action na.action

method

... coxph.control

exact

"clogit""coxph"

[stratacoxphglm](#)


```
## Not run: clogit(case ~ spontaneous + induced + strata(stratum), data=infert)

# A multinomial response recoded to use clogit
# The revised data set has one copy per possible outcome level, with new
# variable tocc = target occupation for this copy, and case = whether
# that is the actual outcome for each subject.
# See the reference below for the data.
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(logan[indx,],
                     id = indx,
                     tocc = factor(rep(resp, each=n)))
logan2$case <- (logan2$occupation == logan2$tocc)
clogit(case ~ tocc + tocc:education + strata(id), logan2)
```

cluster

cluster

cluster(x)

x

x

coxphsurvreg

```
marginal.model <- coxph(Surv(time, status) ~ rx, data= rats, cluster=litter,
                        subset=(sex=='f'))
frailty.model  <- coxph(Surv(time, status) ~ rx + frailty(litter), rats,
                        subset=(sex=='f'))
```

```
colon
```

```
colon  
  data(cancer, package="survival")
```

```
table(colon$nodes, colon$node4)
```

concordance

```
concordance(object, ...)
## S3 method for class 'formula'
concordance(object, data, weights, subset, na.action,
  cluster, ymin, ymax, timewt= c("n", "S", "S/G", "n/G2", "I"),
  influence=0, ranks = FALSE, reverse=FALSE, timefix=TRUE, keepstrata=10, ...)
## S3 method for class 'lm'
concordance(object, ..., newdata, cluster, ymin, ymax,
  influence=0, ranks=FALSE, timefix=TRUE, keepstrata=10)
## S3 method for class 'coxph'
concordance(object, ..., newdata, cluster, ymin, ymax,
  timewt= c("n", "S", "S/G", "n/G2", "I"), influence=0,
  ranks=FALSE, timefix=TRUE, keepstrata=10)
## S3 method for class 'survreg'
concordance(object, ..., newdata, cluster, ymin, ymax,
  timewt= c("n", "S", "S/G", "n/G2", "I"), influence=0,
  ranks=FALSE, timefix=TRUE, keepstrata=10)
```

object	y ~xy ~ x + strata(z)
data	formulasubsetweightsobject
weights	object
subset	object
na.action	options()\\$na.actionobject
...	object
newdata	
cluster	
yminymax	
timewt	
influence	
ranks	
reverse	FALSExyxxcoxphreverse = TRUEcoxph
timefix	
keepstrata	keepstratakeepstrata=FALSE

$$Pr(x_i < x_j | y_i < y_j) x \hat{y}$$

timewt

```
timewttimewt="S"timewt="S/G"timewt="n/G2"timewt= "I"
clogitkeepstrata=FALSEkeepstrata=0keepstrata = 10
```

```
concordance
concordance
count
n
var
cvar          survConcordance
dfbeta
influence
ranks

newdata
```

coxph

```
fit1 <- coxph(Surv(ptime, pstat) ~ age + sex + mspike, mgus2)
concordance(fit1, timewt="n/G2") # Uno's weighting

# logistic regression
tdata <- flchain
options(na.action= na.exclude) # fit2a has many more missings, this causes
# predict to return nrow(flchain) values, but has to be set before the fit
fit2a <- glm(I(sex=='M') ~ age + log(creatinine), binomial, data= flchain)
fit2b <- glm(I(sex=='M') ~ age + kappa + lambda, binomial, data= flchain)
tdata$eta1 <- predict(fit2a)
tdata$eta2 <- predict(fit2b)
cfit <- concordance(I(sex=="M") ~ eta1 + eta2, tdata) # equal to the AUC
coef(cfit) # males and females differ in creatinine, less in kappa/lambda
```

```

convec <- c(1,-1) # compute a contrast
c(test= unname(convec%% coef(cfit)),
  std = unname(sqrt(convec %%% vcov(cfit) %%% convec)))

# compare multiple survival models
options(na.action = na.exclude) # predict all 1384 obs, including missing
fit3 <- glm(pstat ~ age + sex + mspike + offset(log(ptime)),
  poisson, data= mgus2)
fit4 <- coxph(Surv(ptime, pstat) ~ age + sex + mspike, mgus2)
fit5 <- coxph(Surv(ptime, pstat) ~ age + sex + hgb + creat, mgus2)

tdata <- mgus2; tdata$ptime <- 60 # prediction at 60 months
p3 <- -predict(fit3, newdata=tdata)
p4 <- -predict(fit4) # high risk scores predict shorter survival
p5 <- -predict(fit5)
options(na.action = na.omit) # return to the R default

cfit <- concordance(Surv(ptime, pstat) ~p3 + p4 + p5, mgus2)
cfit
round(coef(cfit), 3)
round(cov2cor(vcov(cfit)), 3) # high correlation

test <- c(1, -1, 0) # contrast vector for model 1 - model 2
round(c(difference = test %%% coef(cfit),
  sd= sqrt(test %%% vcov(cfit) %%% test)), 3)

```

concordancefit

concordance

```

concordancefit(y, x, strata, weights, ymin = NULL, ymax = NULL,
  timewt = c("n", "S", "S/G", "n/G2", "I"), cluster, influence = 0,
  ranks = FALSE, reverse = FALSE, timefix = TRUE, keepstrata=10,
  std.err = TRUE)

```

y

x

strata

weights

yminymax

timewt

clusterinfluenceranksreversetimefix

concordance

keepstrata keepstratakeepstrata=FALSE

std.err

concordance

cox.zph

coxph

cox.zph(fit, transform="km", terms=TRUE, singledf=FALSE, global=TRUE)

fit	coxphcoxme
transform	"km""rank""identity"
terms	
singledf	terms=FALSE
global	

xx=TRUEcoxph $\beta(t)\beta(t)$ table
frailtycoxme
ycox.zph

"cox.zph"

table	
x	
time	
strata	coxph model
y	terms
var	
transform	
call	

coxphSurv

```
fit <- coxph(Surv(futime, fustat) ~ age + ecog.ps,
             data=ovarian)
temp <- cox.zph(fit)
print(temp)           # display the results
plot(temp)            # plot curves
```

coxph

```
coxph(formula, data=, weights, subset,
       na.action, init, control,
       ties=c("efron","breslow","exact"),
       singular.ok=TRUE, robust,
       model=FALSE, x=FALSE, y=TRUE, tt, method=ties,
       id, cluster, istate, statedata, nocenter=c(-1, 0, 1), ...)
```

formula	~Surv
data	formulasubsetweights
weights	
subset	
na.action	options()\\$na.action
init	
control	coxph.control coxph.control(...)
ties	
singular.ok	TRUE
robust	clusterid
id	data
cluster	robustdata
istate	data
statedata	
model	TRUEmodel

```
x          TRUEx
y          TRUEy
tt
method     ties
nocenter
...        coxph.control
```

```
nocenter
mean(x)pspline(x)model.frame
```

```
coxphcoxph.objectcoxphms.object
```

```
predictresidualssurvfitcoxphtransform
```

```
tfun <- function(tform) coxph(tform, data=lung)
fit <- tfun(Surv(time, status) ~ age)
predict(fit)
```

```
model=TRUEcoxphfit
```

```
clusterrobust=TRUEerobust=TRUE
```

```
stratattpsplinefrailtyridge+ strata(group)+ cluster(group)cluster
strata
ttid
tt
survival::survival::coxph(survival:Surv(time,      status)      ~      age      +
survival::strata(inst), data=lung)
model.framestats::offset
survival::coxph
```

```
idclusterrobust=TRUEidid
```



```
clogitcoxph
```

```
ties='breslow' ties='efron'  
timefixcoxph.controltimefix=FALSE
```

```
coxph
```

```
coxph.objectcoxphms.objectcoxph.controlclusterstrataSurvsurvfitspline
```

```
# Create the simplest test data set  
test1 <- list(time=c(4,3,1,1,2,2,3),  
             status=c(1,1,1,0,1,1,0),  
             x=c(0,2,1,1,1,0,0),  
             sex=c(0,0,0,0,1,1,1))  
# Fit a stratified model  
coxph(Surv(time, status) ~ x + strata(sex), test1)  
# Create a simple data set for a time-dependent model  
test2 <- list(start=c(1,2,5,2,1,7,3,4,8,8),  
             stop=c(2,3,6,7,8,9,9,9,14,17),  
             event=c(1,1,1,1,1,1,1,0,0,0),  
             x=c(1,0,0,1,0,1,1,1,0,0))  
summary(coxph(Surv(start, stop, event) ~ x, test2))  
  
#  
# Create a simple data set for a time-dependent model  
#  
test2 <- list(start=c(1, 2, 5, 2, 1, 7, 3, 4, 8, 8),  
             stop =c(2, 3, 6, 7, 8, 9, 9, 9,14,17),  
             event=c(1, 1, 1, 1, 1, 1, 1, 0, 0, 0),  
             x      =c(1, 0, 0, 1, 0, 1, 1, 1, 0, 0) )  
  
summary( coxph( Surv(start, stop, event) ~ x, test2))  
  
# Fit a stratified model, clustered on patients  
  
bladder1 <- bladder[bladder$enum < 5, ]  
coxph(Surv(stop, event) ~ (rx + size + number) * strata(enum),  
      cluster = id, bladder1)  
  
# Fit a time transform model using current age  
coxph(Surv(time, status) ~ ph.ecog + tt(age), data=lung,  
      tt=function(x,t,...) pspline(x + t/365.25))
```

coxph.control

coxph

```
coxph.control(eps = 1e-09, toler.chol = .Machine$double.eps^0.75,  
iter.max = 20, toler.inf = sqrt(eps), outer.max = 10, timefix=TRUE)
```

eps

toler.chol

iter.max

toler.inf

outer.max

timefix

tolerance.inf

eps

timefixcoxphsurvfittimefix=TRUEall.equalFALSE

[coxph](#)

coxph.detail

```
coxph.detail(object, riskmat=FALSE, rorder=c("data", "time"))
```

object coxph

riskmat

rorder xyriskmat

```

time
nevent
means      exp(x %*% fit$coef)
nrisk
score
imat
hazard      object$means
varhaz
xy
strata      time
wtrisk
riskmat     coxph

```

[coxphresiduals.coxph](#)

```

fit  <- coxph(Surv(futime,fustat) ~ age + rx + ecog.ps, ovarian, x=TRUE)
fitd <- coxph.detail(fit)
# There is one Schoenfeld residual for each unique death. It is a
# vector (covariates for the subject who died) - (weighted mean covariate
# vector at that time). The weighted mean is defined over the subjects
# still at risk, with exp(X beta) as the weight.

events <- fit$y[,2]==1
etime  <- fit$y[events,1] #the event times --- may have duplicates
indx   <- match(etime, fitd$time)
schoen <- fit$x[events,] - fitd$means[indx,]

```

coxph.object

coxphprintsummaryresidualspredictsurvfit

```

coefficients
var
naive.var      robustvarasympt.var
loglik
score
rscore
wald.test

```

```
iter
linear.predictors
      predict.coxph
residuals
means      predict(fit, type='risk')
n
nevent
n.id      id
concordance
first
weights
method
na.action      na.action
timefix
...      lmtermsassignformulacallxyframe
```

```
coxph
```

[coxphcoxph.detailcox.zphresiduals.coxphsurvfitsurvreg](#)

```
coxph.wtest
```

```
coxph.wtest(var, b, toler.chol = 1e-09)
```

```
var
```

```
b
```

```
toler.chol
```

coxphms.object

coxphcoxph

states

cmap

smap

rmap

rmap

coxph

[coxphcoxph.object](#)

coxsurv.fit

```
coxsurv.fit(ctype, stype, se.fit, varmat, cluster,  
            y, x, wt, risk, position, strata, oldid,  
            y2, x2, risk2, strata2, id2, unlist=TRUE)
```

stype

ctype

se.fit

varmat

cluster

y

x

wt

risk

position

strata

oldid

y2x2risk2strata2

id2 nrow(x2)x2x2

unlist FALSEsurvfit

survfit

survfit.coxphinst/sourcecode.pdf

survfit.coxph

diabetic

```
diabetic
data(diabetic, package="survival")
```

```
id
laser xenonargon
age
eye leftright
trt
risk
time
status
```

```
# juvenile diabetes is defined as and age less than 20
juvenile <- 1*(diabetic$age < 20)
coxph(Surv(time, status) ~ trt + juvenile, cluster= id,
      data= diabetic)
```

dsurvreg

survreg

dsurvreg(x, mean, scale=1, distribution='weibull', parms)
psurvreg(q, mean, scale=1, distribution='weibull', parms)
qsurvreg(p, mean, scale=1, distribution='weibull', parms)
rsurvreg(n, mean, scale=1, distribution='weibull', parms)

x	NA
q	NA
p	NA
n	
mean	pqn
scale	pqn
distribution	survreg.distributions
parms	

qp

locationscalesurvreg

$$\text{survreg} F(t) = 1 - e^{-(\lambda t)^p} \log(t) \alpha, \sigma_{\text{survreg}} \sigma = 1/p \alpha = -\log(\lambda \text{stats::dweibull}(p, 1/\lambda, 1/\sigma) \exp \alpha)$$

dsurvregpsurvregqsurvregmeansd

[survregNormal](#)

```

# List of distributions available
names(survreg.distributions)
## Not run:
[1] "extreme"      "logistic"     "gaussian"     "weibull"      "exponential"
[6] "rayleigh"     "loggaussian"  "lognormal"    "loglogistic"  "t"

## End(Not run)
# Compare results
all.equal(dsurvreg(1:10, 2, 5, dist='lognormal'), dlnorm(1:10, 2, 5))

# Hazard function for a Weibull distribution
x <- seq(.1, 3, length=30)
haz <- dsurvreg(x, 2, 3)/ (1-psurvreg(x, 2, 3))
## Not run:
plot(x, haz, log='xy', ylab="Hazard") #line with slope (1/scale -1)

## End(Not run)

# Estimated CDF of a simple Weibull
fit <- survreg(Surv(time, status) ~ 1, data=lung)
pp <- 1:99/100
q1 <- qsurvreg(pp, coef(fit), fit$scale)
q2 <- qweibull(pp, shape= 1/fit$scale, scale= exp(coef(fit)))
all.equal(q1, q2)
## Not run:
plot(q1, pp, type='l', xlab="Months", ylab="CDF")

## End(Not run)
# per the help page for dweibull, the mean is scale * gamma(1 + 1/shape)
c(mean = exp(coef(fit))* gamma(1 + fit$scale))

```

`finegray`

```

finegray(formula, data, weights, subset, na.action= na.pass, etype,
          prefix="fg", count, id, timefix=TRUE)

```

```

formula
data
weights
subset
na.action
etype
prefix          prefix
count
id
timefix         aeqSurv

```



```
Surv(etime, estat)estat
Surv(entry, exit, stat)id
finegray
clustercoxph
```

[coxphaeqSurv](#)

```
# Treat time to death and plasma cell malignancy as competing risks
etime <- with(mgus2, ifelse(pstat==0, futime, ptime))
event <- with(mgus2, ifelse(pstat==0, 2*death, 1))
event <- factor(event, 0:2, labels=c("censor", "pcm", "death"))

# FG model for PCM
pdata <- finegray(Surv(etime, event) ~ ., data=mgus2)
fgfit <- coxph(Surv(fgstart, fgstop, fgstatus) ~ age + sex,
               weight=fgwt, data=pdata, cluster=id)

# Compute the weights separately by sex
adata <- finegray(Surv(etime, event) ~ . + strata(sex),
                  data=mgus2, na.action=na.pass)
```

flchain

```
flchain
data(flchain, package="survival")
```

age
sex
sample.yr
kappa
lambda
flc.grp
creatinine
mgus
fuptime
death
chapter

```
data(flchain)
age.grp <- cut(flchain$age, c(49,54, 59,64, 69,74,79, 89, 110),
               labels= paste(c(50,55,60,65,70,75,80,90),
                             c(54,59,64,69,74,79,89,109), sep='-'))
table(flchain$sex, age.grp)
```

frailty

```
frailty(x, distribution="gamma", ...)
frailty.gamma(x, sparse = (nclass > 5), theta, df, eps = 1e-05,
              method = c("em","aic", "df", "fixed"), ...)
frailty.gaussian(x, sparse = (nclass > 5), theta, df,
                 method =c("reml","aic", "df", "fixed"), ...)
frailty.t(x, sparse = (nclass > 5), theta, df, eps = 1e-05, tdf = 5,
           method = c("aic", "df", "fixed"), ...)
```

```

x
distribution    gammagaussian frailty.gamma frailty.gaussian frailty.t
...
sparse         x
theta          method='fixed'
df             method='df' thetadf
method         fixeddfaicemreml
tdf
eps

```

```

frailtycoxphsurvreg
survreg
frailtypspine

```

```

coxme

```

```

coxphsurvreg

```

```

# Random institutional effect
coxph(Surv(time, status) ~ age + frailty(inst, df=4), lung)

# Litter effects for the rats data
rfit2a <- coxph(Surv(time, status) ~ rx +
               frailty.gaussian(litter, df=13, sparse=FALSE), rats,
               subset= (sex=='f'))
rfit2b <- coxph(Surv(time, status) ~ rx +
               frailty.gaussian(litter, df=13, sparse=TRUE), rats,
               subset= (sex=='f'))

```

gbsg

gbsg

```
gbsg  
data(cancer, package="survival")
```

```
pid  
age  
meno  
size  
grade  
nodes  
pgr  
er  
hormon  
rfstime  
status
```

[rotterdam](#)

heart

```
heart  
data(heart, package="survival")
```

stanford2

hoel

data("cancer")

trt ControlGerm-free

days

outcome censorthymic lymphomareticulum cell sarcomaother causes

id

```

hsurv <- survfit(Surv(days, outcome) ~ trt, data = hoel, id= id)
plot(hsurv, lty=1:2, col=rep(1:3, each=2), lwd=2, xscale=30.5,
      xlab="Months", ylab= "Death")
legend("topleft", c("Lymphoma control", "Lymphoma germ free",
                    "Sarcoma control", "Sarcoma germ free",
                    "Other control", "Other germ free"),
      col=rep(1:3, each=2), lty=1:2, lwd=2, bty='n')
hfit <- coxph(Surv(days, outcome) ~ trt, data= hoel, id = id)

```

is.ratetable

class

is.ratetable(x, verbose=FALSE)

x

verbose TRUEx

pyearssurvexpverbose

TRUExFALSE

[pyearssurvexp](#)

```

is.ratetable(survexp.us) # True
is.ratetable(lung)      # False

```

kidney

```
kidney
# or
data(cancer, package="survival")
```

```
kfit <- coxph(Surv(time, status)~ age + sex + disease + frailty(id), kidney)
kfit0 <- coxph(Surv(time, status)~ age + sex + disease, kidney)
kfitm1 <- coxph(Surv(time,status) ~ age + sex + disease +
  frailty(id, dist='gauss'), kidney)
```

levels.Surv

Surv

```
## S3 method for class 'Surv'
levels(x)
```

```
x          Surv
```

SurvSurv

```
y1 <- Surv(c(1,5, 9, 17,21, 30),
           factor(c(0, 1, 2,1,0,2), 0:2, c("censored", "progression", "death")))
levels(y1)

y2 <- Surv(1:6, rep(0:1, 3))
y2
levels(y2)
```

lines.survfit

plot.survfit

```
## S3 method for class 'survfit'
lines(x, type="s", pch=3, col=1, lty=1,
      lwd=1, cex=1, mark.time=FALSE, xmax,
      fun, conf.int=FALSE,
      conf.times, conf.cap=.005, conf.offset=.012,
      conf.type = c("log", "log-log", "plain", "logit", "arcsin"),
      mark, noplot="(s0)", cumhaz= FALSE, cumprob= FALSE, ...)
## S3 method for class 'survexp'
lines(x, type="l", ...)
## S3 method for class 'survfit'
points(x, fun, censor=FALSE, col=1, pch,
       noplot="(s0)", cumhaz=FALSE, ...)
```

x	survfitsurvexp
type	linesurvfitsurvexplines.survexplines.survfit
colltylwdcex	points
pch	matplot
mark	pch
censor	points
mark.time	FALSETRUEmark.time
xmax	
fun	fun=loglog=T
conf.int	TRUE"only"
conf.times	
conf.cap	
conf.offset	
conf.type	"plain""log""log-log""logit""none"curve +- k *se(curve)conf.int
noplot	
cumhaz	
cumprob	cumprob
...	


```
survfit
plot.survfitxlimxmaxoptions(plot.survfit) <- NULL
```

```
xycumprob=TRUEyx
```

[linesparplot.survfitsurvfitsurvexp](#)

```
fit <- survfit(Surv(time, status==2) ~ sex, pbc, subset=1:312)
plot(fit, mark.time=FALSE, xscale=365.25,
      xlab='Years', ylab='Survival')
lines(fit[1], lwd=2) #darken the first curve and add marks

# Add expected survival curves for the two groups,
# based on the US census data
# The data set does not have entry date, use the midpoint of the study
efit <- survexp(~sex, data=dbc, times= (0:24)*182, ratetable=survexp.us,
               rmap=list(sex=sex, age=age*365.35, year=as.Date('1979/01/01')))
temp <- lines(efit, lty=2, lwd=2:1)
text(temp, c("Male", "Female"), adj= -.1) #labels just past the ends
title(main="Primary Biliary Cirrhosis, Observed and Expected")
```

logan

```
logan
data(logan, package="survival")
```

farmoperativescraftsmensalesprofessional

non-blackblack

<https://gss.norc.org/>

logLik.coxph

```
## S3 method for class 'coxph'  
logLik(object, ...)  
## S3 method for class 'survreg'  
logLik(object, ...)
```

```
object      coxphsurvreg  
...
```

```
AIC  
survreglogLik.coxph
```

```
logLik
```

[logLik](#)

lung

```
lung  
data(cancer, package="survival")
```

lvcf

lvcf(id, x, time)

id

x

time

timeid

x

na.locfzooLOCFDescToolsidtmerge

tmerge

```
newplat <- with(pbcseq, lvcf(id, platelet))  
table(is.na(pbcseq$platelet), is.na(newplat))
```

mgus

```
mgus  
mgus1  
data(cancer, package="survival")
```

malefemale

10⁶
mgus1

```
# Create the competing risk curves for time to first of death or PCM
sfit <- survfit(Surv(start, stop, event) ~ sex, mgus1, id=id,
               subset=(enum==1))
print(sfit) # the order of printout is the order in which they plot

plot(sfit, xscale=365.25, lty=c(2,2,1,1), col=c(1,2,1,2),
     xlab="Years after MGUS detection", ylab="Proportion")
legend(0, .8, c("Death/male", "Death/female", "PCM/male", "PCM/female"),
     lty=c(1,1,2,2), col=c(2,1,2,1), bty='n')

title("Curves for the first of plasma cell malignancy or death")
# The plot shows that males have a higher death rate than females (no
# surprise) but their rates of conversion to PCM are essentially the same.
```

mgus2

mgus

mgus2
data(cancer, package="survival")

id
age
sex FM
dxyr
hgb
creat
mspike
ptime
pstat
fuptime
death

mgus

model.frame.coxph

```
## S3 method for class 'coxph'  
model.frame(formula, ...)
```

formula	coxph
...	model.frame

residual

[model.frame](#)

`model.matrix.coxph`

```
## S3 method for class 'coxph'
model.matrix(object, data=NULL, contrast.arg =
  object$contrasts, ...)
```

```
object          coxph
data
contrast.arg
...             model.frame
```

```
datamodel.matrix
termsmodel.frame
```

`model.matrix`

```
fit1 <- coxph(Surv(time, status) ~ age + factor(ph.ecog), data=lung)
xfit <- model.matrix(fit1)

fit2 <- coxph(Surv(time, status) ~ age + factor(ph.ecog), data=lung,
              x=TRUE)
all.equal(model.matrix(fit1), fit2$x)
```

myeloid

```
myeloid  
data(cancer, package="survival")
```

```
id  
trt  
sex  
flt3  
fuptime  
death fuptime  
txtime  
crttime  
rltime
```

```
coxph(Surv(fuptime, death) ~ trt + flt3, data=myeloid)  
# See the mstate vignette for a more complete analysis
```

myeloma

```
myeloma  
data("cancer", package="survival")
```



```
id
year
entry
ftime
death
```

```
entry
```

```
# Incorrect survival curve, which ignores left truncation
fit1 <- survfit(Surv(futime, death) ~ 1, myeloma)
# Correct curve
fit2 <- survfit(Surv(entry, futime, death) ~1, myeloma)
```

```
nafld
```

```
nafld1
      nafld2
      nafld3
data(nafld, package="survival")
```

```
nafld1
id
age
male
weight
height
bmi
case.id
ftime
status
nafld2
```

id
days
test
value
nafld3
id
days
event

nafld1nafld2nafld3

neardate

neardate(id1, id2, y1, y2, best = c("after", "prior"),
nomatch = NA_integer_)

id1
id2
y1
y2
best best='prior'best='after'
nomatch

matchfindInterval
id1id2matchid1id2id1id2y1
y1y2c(y1, y2)[as.POSIXct](#)y1y2

nomatch

matchfindInterval

```
data1 <- data.frame(id = 1:10,
                    entry.dt = as.Date(paste("2011", 1:10, "5", sep='-')))
temp1 <- c(1,4,5,1,3,6,9, 2,7,8,12,4,6,7,10,12,3)
data2 <- data.frame(id = c(1,1,1,2,2,4,4,5,5,5,6,8,8,9,10,10,12),
                    lab.dt = as.Date(paste("2011", temp1, "1", sep='-')),
                    chol = round(runif(17, 130, 280)))

#first cholesterol on or after enrollment
indx1 <- neardate(data1$id, data2$id, data1$entry.dt, data2$lab.dt)
data2[indx1, "chol"]

# Closest one, either before or after.
#
indx2 <- neardate(data1$id, data2$id, data1$entry.dt, data2$lab.dt,
                  best="prior")
ifelse(is.na(indx1), indx2, # none after, take before
       ifelse(is.na(indx2), indx1, #none before
              ifelse(abs(data2$lab.dt[indx2]- data1$entry.dt) <
                     abs(data2$lab.dt[indx1]- data1$entry.dt), indx2, indx1)))

# closest date before or after, but no more than 21 days prior to index
indx2 <- ifelse((data1$entry.dt - data2$lab.dt[indx2]) >21, NA, indx2)
ifelse(is.na(indx1), indx2, # none after, take before
       ifelse(is.na(indx2), indx1, #none before
              ifelse(abs(data2$lab.dt[indx2]- data1$entry.dt) <
                     abs(data2$lab.dt[indx1]- data1$entry.dt), indx2, indx1)))
```

nostutter

nostutter(id, x, censor = 0, single=FALSE)

id

x

censor

single

cgdstatus =1nafld
tmerge

censor

lvcftmerge

nsk

nsk(x, df = NULL, knots = NULL, intercept = FALSE, b = 0.05,
Boundary.knots = quantile(x, c(b, 1 - b), na.rm = TRUE))

x
df
knots
intercept
b bs=0ns
Boundary.knots

nsknsintercept = FALSE
nsrms::rCS

ns

```

# make some dummy data
tdata <- data.frame(x= lung$age, y = 10*log(lung$age-35) + rnorm(228, 0, 2))
fit1 <- lm(y ~ -1 + nsk(x, df=4, intercept=TRUE) , data=tdata)
fit2 <- lm(y ~ nsk(x, df=3), data=tdata)

# the knots (same for both fits)
knots <- unlist(attributes(fit1$model[[2]])[c('Boundary.knots', 'knots')])
sort(unname(knots))
unname(coef(fit1)) # predictions at the knot points

unname(coef(fit1)[-1] - coef(fit1)[1]) # differences: yhat[2:4] - yhat[1]
unname(coef(fit2)[-1]) # ditto

## Not run:
plot(y ~ x, data=tdata)
points(sort(knots), coef(fit1), col=2, pch=19)
coef(fit)[1] + c(0, coef(fit)[-1])

## End(Not run)

```

nwtco

```

nwtco
data(nwtco, package="survival")

```

```

seqno
instit
histol
stage
study
rel
edrel
age
in.subcohort

```

```

with(nwtco, table(instit,histol))
anova(coxph(Surv(edrel,rel)~histol+instit,data=nwtco))
anova(coxph(Surv(edrel,rel)~instit+histol,data=nwtco))

```

ovarian

```
ovarian  
data(cancer, package="survival")
```

pbcc

```
pbcc  
data(pbcc, package="survival")
```

pbcsseq

pbcsseq

```
pbcsseq  
data(pbc, package="survival")
```

pbcs

```
# Create the start-stop-event triplet needed for coxph
first <- with(pbcseq, c(TRUE, diff(id) !=0)) #first id for each subject
last  <- c(first[-1], TRUE) #last id

time1 <- with(pbcseq, ifelse(first, 0, day))
time2 <- with(pbcseq, ifelse(last,  futime, c(day[-1], 0)))
event <- with(pbcseq, ifelse(last,  status, 0))

fit1 <- coxph(Surv(time1, time2, event) ~ age + sex + log(bili), pbcseq)
```

plot.aareg

```
## S3 method for class 'aareg'
plot(x, se=TRUE, maxtime, type='s', ...)
```


x aareg
se
maxtime
type
...

plot.cox.zph

```
## S3 method for class 'cox.zph'  
plot(x, resid=TRUE, se=TRUE, df=4, nsmo=40, var,  
      xlab="Time", ylab, lty=1:2, col=1, lwd=1, pch=1, cex=1,  
      hr=FALSE, plot=TRUE, ...)
```

x cox.zph
resid TRUE
se TRUE
df df=2
nsmo
var

hr
xlab
ylab
ltycollwd
plot
pch points
cex points
... plot

coxphcox.zph

```
vfit <- coxph(Surv(time,status) ~ trt + factor(celltype) +
              karno + age, data=veteran, x=TRUE)
temp <- cox.zph(vfit)
plot(temp, var=3)      # Look at Karnofsky score, old way of doing plot
plot(temp[3])          # New way with subscripting
abline(0, 0, lty=3)
# Add the linear fit as well
abline(lm(temp$y[,3] ~ temp$x)$coefficients, lty=4, col=3)
title(main="VA Lung Study")
```

plot.survfit	survfit
--------------	---------

log=T

printcollty

```
## S3 method for class 'survfit'
plot(x, conf.int=, mark.time=FALSE,
     pch=3, col=1, lty=1, lwd=1, cex=1, log=FALSE, xscale=1, yscale=1,
     xlim, ylim, xmax, fun,
     xlab="", ylab="", xaxs="r", conf.times, conf.cap=.005,
     conf.offset=.012,
     conf.type = c("log", "log-log", "plain", "logit", "arcsin", "none"),
     mark, mark.col, noplot="(s0)", cumhaz=FALSE,
     firstx, ymin, cumprob=FALSE, ...)
```

x	survfitsurvfit
conf.int	
mark.time	FALSETRUEmark.time
pch	pointsc("a", "b", "c", "d")mark.time = TRUE
col	
lty	
lwd	
cex	
log	
xscale	yscale
yscale	lines(surv.exp(...))yscale

```

xlimylim
xmax          xlim
fun           fun=logfun=sqrt"S""log"log=T"event""F"  $F(t) = 1 - S(t)$ "cloglog"
              "identity""surv"type="S""cumhaz"

xlab
ylab
xaxs          "S"par"S""i"
conf.times
conf.cap
conf.offset
conf.type     "plain""log""log-log""logit""arcsin""none"curve +- k *se(curve)
conf.int
mark          pch
mark.col
noplots       istate0survcheck
cumhaz        cumhaz
ymin          ylim
firstx        xlim
cumprob       cumprobconf.type="none"
...

```

```

fun='cumhaz'exp(-Λ)ΛΛ
survfitfun
conf.timesconf.offsetconf.offset

```

```

xycumprob=TRUEyx

```

```

xscalesyscalesyscale
survfitcumhaz=TRUEcumhaz=TRUE

```

```

points.survfitlines.survfitparsurvfit

```

```

leukemia.surv <- survfit(Surv(time, status) ~ x, data = aml)
plot(leukemia.surv, lty = 2:3)
legend(100, .9, c("Maintenance", "No Maintenance"), lty = 2:3)
title("Kaplan-Meier Curves\nfor AML Maintenance Study")
lsurv2 <- survfit(Surv(time, status) ~ x, aml, type='fleming')
plot(lsurv2, lty=2:3, fun="cumhaz",
     xlab="Months", ylab="Cumulative Hazard")

```

predict.coxph

coxph

```
## S3 method for class 'coxph'
predict(object, newdata,
  type=c("lp", "risk", "expected", "terms", "survival"),
  se.fit=FALSE, na.action=na.pass, terms=names(object$assign), collapse,
  reference=c("strata", "sample", "zero"), ...)
```

object
newdata
type "lp""risk""expected""terms"
se.fit
na.action newdata
terms
collapse
reference
...

.Machine\$double.max.expcoxphreference="strata"predictreference="sample"
object\$meanslinear.predictorstype="terms""zero"
referencenewdata
frailtynewdata
 $\Lambda(t_i)t_i$ newdata
expectedsurvival

termsmodel=TRUE
model=TRUE

predictcoxphtermplot

```

options(na.action=na.exclude) # retain NA in predictions
fit <- coxph(Surv(time, status) ~ age + ph.ecog + strata(inst), lung)
#lung data set has status coded as 1/2
mresid <- (lung$status-1) - predict(fit, type='expected') #Martingale resid
predict(fit,type="lp")
predict(fit,type="expected")
predict(fit,type="risk",se.fit=TRUE)
predict(fit,type="terms",se.fit=TRUE)

# For someone who demands reference='zero'
pzero <- function(fit)
  predict(fit, reference="sample") + sum(coef(fit) * fit$means, na.rm=TRUE)

```

predict.survreg

survreg

```

## S3 method for class 'survreg'
predict(object, newdata,
  type=c("response", "link", "lp", "linear", "terms", "quantile",
    "uquantile"),
  se.fit=FALSE, terms=NULL, p=c(0.1, 0.9), na.action=na.pass, ...)

```

object	survreg
newdata	
type	"linear""lp""quantile""uquantile""terms""link""lp"
se.fit	TRUE
terms	"terms"
p	
na.action	newdata
...	

[survregresiduals.survreg](#)

```

# Draw figure 1 from Escobar and Meeker, 1992.
fit <- survreg(Surv(time,status) ~ age + I(age^2), data=stanford2,
  dist='lognormal')
with(stanford2, plot(age, time, xlab='Age', ylab='Days',
  xlim=c(0,65), ylim=c(.1, 10^5), log='y', type='n'))
with(stanford2, points(age, time, pch=c(2,4)[status+1], cex=.7))
pred <- predict(fit, newdata=list(age=1:65), type='quantile',
  p=c(.1, .5, .9))
matlines(1:65, pred, lty=c(2,1,2), col=1)

# Predicted Weibull survival curve for a lung cancer subject with
# ECOG score of 2
lfit <- survreg(Surv(time, status) ~ ph.ecog, data=lung)
pct <- 1:98/100 # The 100th percentile of predicted survival is at +infinity
ptime <- predict(lfit, newdata=data.frame(ph.ecog=2), type='quantile',
  p=pct, se=TRUE)
matplot(cbind(ptime$fit + 2*ptime$se.fit,
  ptime$fit - 2*ptime$se.fit)/30.5, 1-pct,
  xlab="Months", ylab="Survival", type='l', lty=c(1,2,2), col=1)

```

```
print.aareg
```

```

## S3 method for class 'aareg'
print(x, maxtime, test=c("aalen", "nrisk"),scale=1,...)

```

```

x          aareg
maxtime
test
scale
...

```

```
print.summary.coxph
```

```
## S3 method for class 'summary.coxph'
print(x, digits=max(getOption("digits") - 3, 3),
      signif.stars = getOption("show.signif.stars"), expand=FALSE, ...)
## S3 method for class 'summary.coxph.penal'
print(x, digits=max(getOption("digits") - 3, 3),
      signif.stars = getOption("show.signif.stars"), maxlabel=25, ...)
```

```
x                summary.coxph
digits
signif.stars
expand
maxlabel
...
```

```
print.summary.survexp
```

```
summary.survexp
```

```
## S3 method for class 'summary.survexp'
print(x, digits = max(options()$digits - 4, 3), ...)
```

```
x                summary.survexp
digits
...
```

```
x
```

```
link{summary.survexp}survexp
```

```
print.summary.survfit
```

```
summary.survfit
```

```
## S3 method for class 'summary.survfit'
print(x, digits = max(options() $digits-4, 3), ...)
```

```
x          "summary.survfit"summary.survfit
digits
...
```

```
x
```

```
summary.survfit
```

```
optionsprintsummary.survfit
```

```
print.survfit
```

```
## S3 method for class 'survfit'
print(x, scale=1, digits = max(options()$digits - 4,3),
      print.rmean=getOption("survfit.print.rmean"),
      rmean = getOption('survfit.rmean'),...)
```

```
x          survfit
scale      scale=365
digits
print.rmeanrmean
```

```
...
```

```
rmeanrmean=365"none""common""individual""common""individual"
```


tablesummary.survfit

rmean

summary.survfitquantile.survfit

pseudo

pseudo(fit, times, type, collapse= TRUE, data.frame=FALSE, ...)

fit	survfit
times	
type	pstatecumhazsojourn
collapse	id
data.frame	
...	residuals.survfitweighted

typesurvstatechazcumhazrmstrmtsaucojourn
survfitsurvfitidsurvfit

fit\$statescollapse=TRUE
survfitidididid = patno

model=TRUEsurvfit

residuals.survfit

```
fit1 <- survfit(Surv(time, status) ~ 1, data=lung)
yhat <- pseudo(fit1, times=c(365, 730))
dim(yhat)
lfit <- lm(yhat[,1] ~ ph.ecog + age + sex, data=lung)

# Restricted Mean Time in State (RMST)
rms <- pseudo(fit1, times= 730, type='RMST') # 2 years
rfit <- lm(rms ~ ph.ecog + sex, data=lung)
rhat <- predict(rfit, newdata=expand.grid(ph.ecog=0:3, sex=1:2), se.fit=TRUE)
# print it out nicely
temp1 <- cbind(matrix(rhat$fit, 4,2))
temp2 <- cbind(matrix(rhat$se.fit, 4, 2))
temp3 <- cbind(temp1[,1], temp2[,1], temp1[,2], temp2[,2])
dimnames(temp3) <- list(paste("ph.ecog", 0:3),
                        c("Male RMST", "(se)", "Female RMST", "(se)"))

round(temp3, 1)
# compare this to the fully non-parametric estimate
fit2 <- survfit(Surv(time, status) ~ ph.ecog, data=lung)
print(fit2, rmean=730)
# the estimate for ph.ecog=3 is very unstable (n=1), pseudovalues smooth it.
#
# In all the above we should be using the robust variance, e.g., svyglm, but
# a recommended package can't depend on external libraries.
# See the vignette for a more complete exposition.
```

pspline

```
pspline(x, df=4, theta, nterm=2.5 * df, degree=3, eps=0.1, method,
        Boundary.knots=range(x), intercept=FALSE, penalty=TRUE, combine, ...)
```

```
psplineinverse(x)
```

```
x
df          dftheta df=0
theta
nterm
degree
eps         df
method      theta
```

```

...
Boundary.knots
intercept
penalty
combine          combine

pspline, coxph.penalty

```

coxphsurvregridgefrailty

```

lfit6 <- survreg(Surv(time, status)~pspline(age, df=2), lung)
plot(lung$age, predict(lfit6), xlab='Age', ylab="Spline prediction")
title("Cancer Data")
fit0 <- coxph(Surv(time, status) ~ ph.ecog + age, lung)
fit1 <- coxph(Surv(time, status) ~ ph.ecog + pspline(age,3), lung)
fit3 <- coxph(Surv(time, status) ~ ph.ecog + pspline(age,8), lung)
fit0
fit1
fit3

```

pyears

```

pyears(formula, data, weights, subset, na.action, rmap,
        ratetable, scale=365.25, expect=c('event', 'pyears'),
        model=FALSE, x=FALSE, y=FALSE, data.frame=FALSE)

```

```

formula          Survsurvfittcutratetable
data              formulasubsetweights
weights
subset
na.action        subsetoptions()$na.action
rmap
ratetable        survexp.uswhite
scale
expect
data.frame
modelxy

```

```
pyears
```

```
py <- pyears(futime ~ rx, rmap=list(age=age, sex=sex, year=entry.dt),  
             ratetable=survexp.us)
```

```
futimeageentry.dt
```

```
rmapsurvexp.ussummary{survexp.us}agesexyearrmapmydata
```

```
tcutcut(age, c(0,2,10,25,100))tcutcutpyears
```

```
pyearsprint
```

```
pyears
```

```
n          pyears
```

```
event      Surv
```

```
expected   expect ="pyears"ratetable
```

```
data       data.frameeventpyearsevent
```

```
offtable   pyears
```

```
tcut
```

```
summary
```

```
call
```

```
observations
```

```
na.action  na.actionna.action
```

```
ratetablesurvexpSurv
```

```
# Look at progression rates jointly by calendar date and age
```

```
#
```

```
temp.yr <- tcut(mgus$dxyr, 55:92, labels=as.character(55:91))
```

```
temp.age <- tcut(mgus$age, 34:101, labels=as.character(34:100))
```

```
pctime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
```

```
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
```

```
pfit <- pyears(Surv(pctime/365.25, pstat) ~ temp.yr + temp.age + sex, mgus,  
               data.frame=TRUE)
```

```
# Turn the factor back into numerics for regression
```

```
tdata <- pfit$data
```

```
tdata$age <- as.numeric(as.character(tdata$temp.age))
```

```
tdata$year <- as.numeric(as.character(tdata$temp.yr))
```

```
fit1 <- glm(event ~ year + age + sex + offset(log(pyears)),  
            data=tdata, family=poisson)
```

```
## Not run:
```

```
# fit a gam model
```

```
gfit.m <- gam(y ~ s(age) + s(year) + offset(log(time)),  
              family = poisson, data = tdata)
```

```

## End(Not run)

# Example #2 Create the hearta data frame:
hearta <- by(heart, heart$id,
            function(x)x[x$stop == max(x$stop),])
hearta <- do.call("rbind", hearta)
# Produce pyears table of death rates on the surgical arm
# The first is by age at randomization, the second by current age
fit1 <- pyears(Surv(stop/365.25, event) ~ cut(age + 48, c(0,50,60,70,100)) +
              surgery, data = hearta, scale = 1)
fit2 <- pyears(Surv(stop/365.25, event) ~ tcut(age + 48, c(0,50,60,70,100)) +
              surgery, data = hearta, scale = 1)
fit1$event/fit1$pyears #death rates on the surgery and non-surg arm

fit2$event/fit2$pyears #death rates on the surgery and non-surg arm

```

quantile.survfit

```

## S3 method for class 'survfit'
quantile(x, probs = c(0.25, 0.5, 0.75), conf.int = TRUE,
         scale, tolerance= sqrt(.Machine$double.eps), ...)
## S3 method for class 'survfitms'
quantile(x, probs = c(0.25, 0.5, 0.75), conf.int = TRUE,
         scale, tolerance= sqrt(.Machine$double.eps), ...)
## S3 method for class 'survfit'
median(x, ...)

```

```

x
probs
conf.int
scale          scale=365.25
tolerance
...

```

```

prob
conf.typesurvfit
tolerance

```

```

survfit
quantilelowerupper

```

survfitprint.survfitqsurvreg

```
fit <- survfit(Surv(time, status) ~ ph.ecog, data=lung)
quantile(fit)

cfits <- coxph(Surv(time, status) ~ age + strata(ph.ecog), data=lung)
csurv<- survfit(cfits, newdata=data.frame(age=c(40, 60, 80)),
               conf.type ="none")
temp <- quantile(csurv, 1:5/10)
temp[2,3,] # quantiles for second level of ph.ecog, age=80
quantile(csurv[2,3], 1:5/10) # quantiles of a single curve, same result
```

ratetable

ratetable(...)

...

rmap

ratetableDate

ratetable

ratetableDate(x)

x

survexppyears

[pyearssurvexp](#)

ratetables

```
data(survexp, package="survival")
```

```
−log(1 − q)/365.25qp = 1 − q  
ratetable  
survexp.us
```

[ratetablesurvexppyears](#)

```
survexp.uswhite <- survexp.usr[,,"white",]
```

rats

```
rats  
data(cancer, package="survival")
```

rats2

```
rats2  
data(cancer, package="survival")
```

reliability

```
data(reliability, package="survival")
```

```
braking
capacitor
  temperature
  voltage
  time
  status
cracks
```

```
genfan
  hours
  status
ifluid
  time
  voltage
imotor
```

```
turbine
```

```
valveSeat
```

```

survreg(Surv(time, status) ~ temperature + voltage, capacitor)

# Figure 16.7 of Meeker, cumulative replacement of locomotive braking
# grids
gfit <- survfit(Surv(day1, day2, status) ~ batch, braking, id= locomotive)
plot(gfit, cumhaz=TRUE, col=1:2, xscale=30.5, conf.time= c(6,12,18)*30.5,
     xlab="Locomotive Age in Months",
     ylab="Mean cumulative number of replacements")

# Replacement of valve seats. In this case the cumulative hazard is the
# natural target, an estimate of the number of replacements by a given time
# (known as the cumulative mean function = CMF in reliability).
# When two valve seats failed at the same inspection, we need to jitter one
# of the times, to avoid a (time1, time2) interval of length 0
ties <- which(with(valveSeat, diff(id)==0 & diff(time)==0)) #first of a tie
temp <- valveSeat$time
temp[ties] <- temp[ties] - .1 # jittered time
vdata <- valveSeat
vdata$time1 <- ifelse(!duplicated(vdata$id), 0, c(0, temp[-length(temp)]))
vdata$time2 <- temp
fit2 <- survfit(Surv(time1, time2, status) ~1, vdata, id=id)
## Not run:
plot(fit2, cumhaz= TRUE, xscale= 365.25,
     xlab="Years in service", ylab = "Expected number of repairs")

## End(Not run)

```

residuals.coxph

```

## S3 method for class 'coxph'
residuals(object,
  type=c("martingale", "deviance", "score", "schoenfeld",
        "dfbeta", "dfbetas", "scaledsch", "partial"),
  collapse=FALSE, weighted= (type %in% c("dfbeta", "dfbetas")),
  na.action, ...)
## S3 method for class 'coxphms'
residuals(object,
  type=c("martingale", "score", "schoenfeld",
        "dfbeta", "dfbetas", "scaledsch"),
  collapse=FALSE, weighted= (type %in% c("dfbeta", "dfbetas")),
  na.action, ...)
## S3 method for class 'coxph.null'
residuals(object,
  type=c("martingale", "deviance", "score", "schoenfeld"),
  collapse=FALSE, weighted= FALSE, ...)

```

```

object      coxphcoxph
type        "martingale""deviance""score""schoenfeld""dfbetas""scaledsch"
            "partial"
collapse
weighted    TRUE
na.action    "na.omit""na.exclude" coxphcoxphcollapse=TRUE
...

```

```
collapse=FALSE clusterid coxphcollapse
```

```
dfbetadfbetasmale_1:2male_1:2
stratatransition
```

[coxphpredict.coxph](#)

```

fit <- coxph(Surv(start, stop, event) ~ (age + surgery)* transplant,
             data=heart)
mresid <- resid(fit, collapse=heart$id)

```

```
residuals.survfit
```

```

## S3 method for class 'survfit'
residuals(object, times,
          type="pstate", collapse=FALSE, weighted= collapse, data.frame=FALSE,
          extra = FALSE, ...)

```

```

object      survfit
times
type
collapse
weighted
data.frame
extra        data.frame=FALSE
...

```

```
pseudo
survfit
pstatesurvvcumhazchazsojournrmstrmtsauc
collapse=TRUEidfit
```

```
times
```

```
idsurvfitsurvfit(... id= abc$def[z])id
```

```
survfitsurvfit.formula
```

```
fit <- survfit(Surv(time, status) ~ x, aml)
resid(fit, times=c(24, 48), type="RMTS")
```

```
residuals.survreg
```

```
residuals survreg
```

```
## S3 method for class 'survreg'
residuals(object, type=c("response", "deviance", "dfbeta", "dfbetas",
"working", "ldcase", "ldresp", "ldshape", "matrix"), rsigma=TRUE,
collapse=FALSE, weighted=FALSE, ...)
```

```
object      survreg
type        "response""deviance""dfbeta""dfbetas""working""ldcase""lsresp"
           "ldshape""matrix"
rsigma
collapse
weighted
...
```

```
 $LpX\beta s\log(\sigma)LdL/dp\partial^2L/\partial p^2dL/ds\partial^2L/\partial s^2\partial^2L/\partial p\partial sldcaseldrespshape$ 
loglik
```

`predict.survreg`

```
fit <- survreg(Surv(futime, death) ~ age + sex, mgus2)
summary(fit) # age and sex are both important

rr <- residuals(fit, type='matrix')
sum(rr[,1]) - with(mgus2, sum(log(futime[death==1]))) # loglik

plot(mgus2$age, rr[,2], col= (1+mgus2$death)) # ldresp
```

`retinopathy`

```
retinopathy
data(retinopathy, package="survival")
```

```
id
laser xenonargon
eye rightleft
age
type juvenileadult
trt
futime
status
risk
```

```
coxph(Surv(futime, status) ~ type + trt, cluster= id, retinopathy)
```

rhDNase

```
rhDNase
data(rhDNase, package="survival")
```

```
id
inst
trt
entry.dt
end.dt
fev
ivstart
ivstop
```

```
# Build the start-stop data set for analysis, and
# replicate line 2 of table 8.13 in the book
first <- subset(rhDNase, !duplicated(id)) #first row for each subject
dnase <- tmerge(first, first, id=id, tstop=as.numeric(end.dt -entry.dt))

# Subjects whose fu ended during the 6 day window are the reason for
# this next line
temp.end <- with(rhDNase, pmin(ivstop+6, end.dt-entry.dt))
dnase <- tmerge(dnase, rhDNase, id=id,
               infect=event(ivstart),
               end= event(temp.end))
# toss out the non-at-risk intervals, and extra variables
# 3 subjects had an event on their last day of fu, infect=1 and end=1
dnase <- subset(dnase, (infect==1 | end==0), c(id:trt, fev:infect))
agfit <- coxph(Surv(tstart, tstop, infect) ~ trt + fev, cluster=id,
               data=dnase)
```

ridge

thetascale=Tdftheta

ridge(..., theta, df=nvar/2, eps=0.1, scale=TRUE)

...

theta theta

df

eps df

scale

coxph.penalty

ridge(x1, x2, x3, ...)ridgemdata\$many <- as.matrix(mydata[,5:53])

[coxphsurvregpsplinefrailty](#)

coxph(Surv(futime, fustat) ~ rx + ridge(age, ecog.ps, theta=1),
 ovarian)

lfit0 <- survreg(Surv(time, status) ~1, lung)

lfit1 <- survreg(Surv(time, status) ~ age + ridge(ph.ecog, theta=5), lung)

lfit2 <- survreg(Surv(time, status) ~ sex + ridge(age, ph.ecog, theta=1), lung)

lfit3 <- survreg(Surv(time, status) ~ sex + age + ph.ecog, lung)

rotterdam

rotterdam

rotterdam
data(cancer, package="survival")

pid
year
age
meno
size <=2020-50>50
grade
nodes
pgr
er
hormon
chemo
rtime
recur
dtime
death

[gbsg](#)


```

# liberal definition of rfs (count later deaths)
rfs <- pmax(rotterdam$recur, rotterdam$death)
rfstime <- with(rotterdam, ifelse(recur==1, rtime, dtime))
fit1 <- coxph(Surv(rfstime, rfs) ~ pspline(age) + meno + size +
              pspline(nodes) + er, data = rotterdam)

# conservative (no deaths after last fu for recurrence)
ignore <- with(rotterdam, recur ==0 & death==1 & rtime < dtime)
table(ignore)
rfs2 <- with(rotterdam, ifelse(recur==1 | ignore, recur, death))
rfstime2 <- with(rotterdam, ifelse(recur==1 | ignore, rtime, dtime))
fit2 <- coxph(Surv(rfstime2, rfs2) ~ pspline(age) + meno + size +
              pspline(nodes) + er, data = rotterdam)

# Note: Both age and nodes show non-linear effects.
# Royston and Altman used fractional polynomials for the nonlinear terms

```

royston

```
royston(fit, newdata, ties = TRUE, adjust = FALSE)
```

```

fit
newdata
ties
adjust

```

```
R.DDR.KOR.NC.GH
```

```
summary.coxphrsqnewdata
```

```

# An example used in Royston and Sauerbrei
pbc2 <- na.omit(pbc) # no missing values
cf1t <- coxph(Surv(time, status==2) ~ age + log(bili) + edema + albumin +
              stage + copper, data=pbc2, ties="breslow")
royston(cf1t)

```

rttright

```
rttright(formula, data, weights, subset, na.action, times, id, timefix = TRUE,
         renorm= TRUE)
```

formula	Surv~
data	subsetweights
weights	subset
subset	
na.action	subsetoptions()\$na.action
times	
id	
timefix	
renorm	

```
formulasurvfit
time
```

survfit

```
afit <- survfit(Surv(time, status) ~1, data=aml)
rwt  <- rttright(Surv(time, status) ~1, data=aml)

# Reproduce a Kaplan-Meier
index <- order(aml$time)
cdf <- cumsum(rwt[index]) # weighted CDF
cdf <- cdf[!duplicated(aml$time[index], fromLast=TRUE)] # remove duplicate times
cbind(time=afit$time, KM= afit$surv, RTTR= 1-cdf)

# Hormonal patients have a different censoring pattern
wt2 <- rttright(Surv(dtime, death) ~ hormon, rotterdam, times= 365*c(3, 5))
dim(wt2)
```

solder

```
solder
data(solder, package="survival")
```

Opening

Solder

Mask

PadType

Panel

skips

```
fit1 <- glm(skips ~ Opening * Solder, poisson, solder,
            subset= (Mask != "A6"))
anova(fit1) # The interaction is important
dummy <- expand.grid(Opening= c("S", "M", "L"), Solder=c("Thin", "Thick"))
yhat <- matrix(predict(fit1, newdata=dummy), ncol=2,
               dimnames=list(Opening= c("S", "M", "L"), Solder=c("Thin", "Thick")))
yhat <- cbind(yhat, difference= yhat[,1]- yhat[,2])
round(yhat, 1) # thin and thick have different patterns

# The balanced subset used by Chambers and Hastie
# contains the first 180 of each mask and deletes mask A6.
index <- 1 + (1:nrow(solder)) - match(solder$Mask, solder$Mask)
solder.balance <- droplevels(subset(solder, Mask != "A6" & index <= 180))
```

stanford2

heart

stanford2

[predict.survregheart](#)

statefig

```
statefig(layout, connect, margin = 0.03, box = TRUE, cex = 1, col = 1,  
         lwd=1, lty=1, bcol=col, acol=col, alwd=lwd, alty=lty, offset=0)
```

layout

connect connect[i,j] !=0

margin

box

cexcolltylwd

bcol

acolalwdalty

offset

```
connect
layoutlayout
connect
boxmargin
```

```
layoutdiagram
```

```
# Draw a simple competing risks figure
states <- c("Entry", "Complete response", "Relapse", "Death")
connect <- matrix(0, 4, 4, dimnames=list(states, states))
connect[1, -1] <- c(1.1, 1, 0.9)
statefig(c(1, 3), connect)
```

```
strata
```

```
strata(..., na.group=FALSE, shortlabel, sep=', ')
```

```
...
na.group      TRUE
shortlabel    TRUE
sep
```

```
coxphinteractionstrata
```

```
coxphinteraction
```

```
a <- factor(rep(1:3,4), labels=c("low", "medium", "high"))
b <- factor(rep(1:4,3))
levels(strata(b))
levels(strata(a,b,shortlabel=TRUE))

coxph(Surv(futime, fustat) ~ age + strata(rx), data=ovarian)
```

summary.aareg

```
## S3 method for class 'aareg'
summary(object, maxtime, test=c("aalen", "nrisk"), scale=1,...)
```

```
object          aareg
maxtime
test
scale
...
```

```
maxtime
```

```
table
test
test.statistic
test.var
test.var2
chisq
n
```

```
afit <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
              dfbeta=TRUE)
summary(afit)
## Not run:
```

	slope	test	se(test)	robust se	z	p
Intercept	5.05e-03	1.9	1.54	1.55	1.23	0.219000
age	4.01e-05	108.0	109.00	106.00	1.02	0.307000
sex	-3.16e-03	-19.5	5.90	5.95	-3.28	0.001030
ph.ecog	3.01e-03	33.2	9.18	9.17	3.62	0.000299

```
Chisq=22.84 on 3 df, p=4.4e-05; test weights=aalen
```

```
## End(Not run)

summary(afit, maxtime=600)
## Not run:
      slope  test se(test) robust se      z      p
Intercept 4.16e-03  2.13    1.48    1.47  1.450 0.146000
      age  2.82e-05 85.80   106.00   100.00  0.857 0.392000
      sex -2.54e-03 -20.60    5.61    5.63 -3.660 0.000256
      ph.ecog 2.47e-03 31.60    8.91    8.67  3.640 0.000271

Chisq=27.08 on 3 df, p=5.7e-06; test weights=aalen

## End(Not run)
```

```
summary.coxph
```

```
## S3 method for class 'coxph'
summary(object, conf.int=0.95, scale=1,...)
```

```
object
conf.int
scale
...
```

```
summary.coxph
nnevent
loglik
coefficients
conf.int
logtestsctestwaldtest
```

```
concordance
used.robust
rsq
fail
call
na.action
```

```
royston
```

[coxphprint.coxph](#)

```
fit <- coxph(Surv(time, status) ~ age + sex, lung)
summary(fit)
```

summary.pyyears

```
## S3 method for class 'pyyears'
summary(object, header = TRUE, call = header, n = TRUE,
event = TRUE, pyyears = TRUE, expected = TRUE, rate = FALSE, rr =expected,
ci.r = FALSE, ci.rr = FALSE, totals=FALSE, legend = TRUE, vline = FALSE,
vertical= TRUE, nastring=".", conf.level = 0.95,
scale = 1, ...)
```

```
object
header
call
neventpyearsexpected
```

```
rateci.r
rrci.rr
totals
legend
vline
vertical
nastring
conf.level
scale
...          format
```

```
pyyears
pyyearsyears
tcut
```


[cipoissonyearsformat](#)

`summary.survexp`

```
## S3 method for class 'survexp'  
summary(object, times, scale = 1, ...)
```

```
object      survexp  
times  
scale      survfitscale = 365.25  
...
```

```
surv  
time  
n.risk
```

[survexp](#)

summary.survfit

```
## S3 method for class 'survfit'
summary(object, times, censored=FALSE, scale=1,
  extend=FALSE, rmean=getOption('survfit.rmean'), data.frame=FALSE, dosum, ...)
## S3 method for class 'survfitms'
summary(object, times, censored=FALSE, scale=1,
  extend=FALSE, rmean=getOption('survfit.rmean'), data.frame=FALSE, dosum, ...)
```

```
object      survfit
times       censored=Ttimesfittimes
censored    times
scale       survfitscale = 365.25
extend      timestimestimes
rmean       print.survfit
data.frame
dosum       times
...
```

```
data.frame = TRUE
data.frame = FALSE
```

```
surv
time
n.risk      survfit
n.event     times
n.entered   times
n.exit.censored
            times

std.err
conf.int
lower
upper
strata      strataNULLsurvtimen.riskstrata
call        fit
na.action   fit
table       print.survfit
type
```

```

extend=FALSEtimesextend=TRUEdata.frame = TRUE
timesSurv(c(0,0, 1, 5), c(2, 3, 8, 10), c(1, 0, 1, 0))(0,2] (0, 3+] (1, 8] (5,10+]
survfitsummarysurvfit
dosum = TRUEdosum=FALSEtimesdosumtimes
survpstate

```

[survfitprint.summary.survfit](#)

```

summary( survfit( Surv(futime, fustat)~1, data=ovarian))
summary( survfit( Surv(futime, fustat)~rx, data=ovarian))

```

Surv

```

Surv(time, time2, event,
      type=c('right', 'left', 'interval', 'counting', 'interval2'),
      origin=0)
is.Surv(x)

```

```

time
event      TRUEFALSETRUEtime
time2      (start, end]event
type       "right""left""counting""interval""interval2"eventtime2
origin
x

```

```

type = "interval"time2
survregsurvfit
if (max(status)==2)Surv(time, status==3)
x[1:3]dropdrop=Fx[1:3,,drop=F]

```

```

Survprintis.naSurvinputAttributes
is.SurvTRUEx"Surv"FALSE

```

Survlung

coxphsurvfitsurvreglung

```
with(aml, Surv(time, status))
survfit(Surv(time, status) ~ ph.ecog, data=lung)
Surv(heart$start, heart$stop, heart$event)
```

Surv-methods

Surv

```
## S3 method for class 'Surv'
anyDuplicated(x, ...)
## S3 method for class 'Surv'
as.character(x, ...)
## S3 method for class 'Surv'
as.data.frame(x, ...)
## S3 method for class 'Surv'
as.matrix(x, ...)
## S3 method for class 'Surv'
c(...)
## S3 method for class 'Surv'
duplicated(x, ...)
## S3 method for class 'Surv'
format(x, ...)
## S3 method for class 'Surv'
head(x, ...)
## S3 method for class 'Surv'
is.na(x)
## S3 method for class 'Surv'
length(x)
## S3 method for class 'Surv'
mean(x, ...)
## S3 method for class 'Surv'
median(x, na.rm=FALSE, ...)
## S3 method for class 'Surv'
names(x)
## S3 replacement method for class 'Surv'
names(x) <- value
## S3 method for class 'Surv'
quantile(x, probs, na.rm=FALSE, ...)
## S3 method for class 'Surv'
plot(x, ...)
## S3 method for class 'Surv'
rep(x, ...)
## S3 method for class 'Surv'
rep.int(x, ...)
```

```

    ## S3 method for class 'Surv'
rep_len(x, ...)
    ## S3 method for class 'Surv'
rev(x)
    ## S3 method for class 'Surv'
t(x)
    ## S3 method for class 'Surv'
tail(x, ...)
    ## S3 method for class 'Surv'
unique(x, ...)

```

```

x          Surv
probs
na.rm
value      xNULL
...

```

Surv

```

as.character
printformatas.character
lengthSurvx <- Surv(1:4, 5:8, c(1,0,1,0)); length(x)names(x)rownames
levels
medianquantileplotsurvfit
xtfrm
unique
c()c(Surv(1:4),    Surv(5:6))c(Surv(1:4),    5:6)c(5:6,    Surv(1:4))c(5:6,
as.vector(Surv(1:4)))

```

Surv

Surv2

Surv2(time, event, repeated=FALSE)

```

time
event
repeated

```

coxphsurvfit

repeated

Surv2printis.na

[Surv2datacoxphsurvfit](#)

Surv2data

coxphsurvfitSurv2

Surv2data(formula, data, subset, id)

formula

data

subset

id

mf

S2.y

S2.state

survcheck

```
survcheck(formula, data, subset, na.action, id, istate, istate0="(s0)",
timefix=TRUE,...)
```

formula	Surv
data	idistate
subset	
na.action	options()\\$na.action
id	
istate	
istate0	istate
timefix	aeqSurv
...	weights

y

flagsSurv
survfitcoxph

states	istate
transitions	
statecount	
flags	
istate	
overlap	
gaps	
teleport	
jumps	

```
xSurv(c(0, 10, a), c(10, 20, censor), c(20,0,c))istatetdctmerge
```

```
istatesurvcheckistateistate
```

survcondense

```
survcondense(formula, data, subset, weights, na.action= na.pass, id,
              start = "tstart", end = "tstop", event = "event")
```

```
formula      ~Surv
data          formulaid
subset
weights
na.action
id
start
end
event
```

```
survSplittmergeformula
```

```
startstopendSurv(time1, time2, death | progression)event
```

survSplittmerge

```
dim(aml)
test1 <- survSplit(Surv(time, status) ~ ., data=aml,
                   cut=c(10, 20, 30), id="newid")
dim(test1)

# remove the added rows
test2 <- survcondense(Surv(tstart, time, status) ~ x, test1, id=newid)
dim(test2)
```

survdiff

G^p

survdiff(formula, data, subset, na.action, rho=0, timefix=TRUE)

formula	Surv(time, status) ~ predictoroffset(sp)spstratastrata na.group=T
data	
subset	
na.action	model.frameoptions()\$na.action
rho	
timefix	aeqSurv

n
obs
exp
chisq
var
strata
pvalue

$S(t)^{\rho}S(t)_{\rho} = 0_{\rho} = 1$

factorexclude

```
## Two-sample test
survdif(Surv(futime, fustat) ~ rx,data=ovarian)
check <- coxph(Surv(futime, fustat) ~ rx, data=ovarian, ties="breslow")
round(summary(check)$sctest, 3)

## Stratified 8-sample test (7 df)
survdif(Surv(time, status) ~ pat.karno + strata(inst), data=lung)
check <- coxph(Surv(time, status) ~ factor(pat.karno) + strata(inst), lung)
round(summary(check)$sctest, 3)

## Expected survival for heart transplant patients based on
## US mortality tables
expect <- survexp(futime ~ 1, data=jasa, cohort=FALSE,
                  rmap= list(age=(accept.dt - birth.dt), sex=1, year=accept.dt),
                  ratetable=survexp.us)
## actual survival is much worse (no surprise)
survdif(Surv(jasa$futime, jasa$fustat) ~ offset(expect))

# The free light chain data set is close to the population.
e2 <- survexp(futime ~ 1, data=flchain, cohort=FALSE,
              rmap= list(age= age*365.25, sex=sex,
                          year=as.Date(paste0(sample.yr, "-07-01"))),
              ratetable= survexp.mn)
survdif(Surv(futime, death) ~ offset(e2), flchain)
```

survexp

```
survexp(formula, data, weights, subset, na.action, rmap, times,
        method=c("ederer", "hakulinen", "conditional", "individual.h",
                  "individual.s"),
        cohort=TRUE, conditional=FALSE,
        ratetable=survival::survexp.us, scale=1,
        se.fit, model=FALSE, x=FALSE, y=FALSE)
```

formula	+survfit~1
data	formulasubsetweights
weights	
subset	data
na.action	subsetoptions()\$na.action
rmap	
times	formula
method	individualindividual.sindividual.hmethod='ederer' method='hakulinen'

```

cohort      methodmethod='individual.s'
conditional methodmethod='conditional'
ratetable   survexp.mnsurvival::
scale       ratetablescale = 365.25
se.fit
modelxy

```

```
survexp.us
```

```

haz <- survexp(fu.time ~ 1, data=mydata,
               rmap = list(year=entry.dt, age=(birth.dt-entry.dt)),
               method='individual.h'))

```

```

glm(status ~ 1 + offset(log(haz)), family=poisson)
glm(status ~ x + offset(log(haz)), family=poisson)

```

```
x
```

```
rmapsurvexp.ussummary{survexp.us}agesexyearrmapmydata
```

```
cohort=TRUEsurvexp
```

[survfityearssurvexp.usratetablesurvexp.fit](#)

```

#
# Stanford heart transplant data
# We don't have sex in the data set, but know it to be nearly all males.
# Estimate of conditional survival
fit1 <- survexp(futime ~ 1, rmap=list(sex="male", year=accept.dt,
                                     age=(accept.dt-birth.dt)), method='conditional', data=jasa)
summary(fit1, times=1:10*182.5, scale=365) #expected survival by 1/2 years

# Estimate of expected survival stratified by prior surgery
survexp(~ surgery, rmap= list(sex="male", year=accept.dt,
                              age=(accept.dt-birth.dt)), method='ederer', data=jasa,
        times=1:10 * 182.5)

```

```
## Compare the survival curves for the Mayo PBC data to Cox model fit
##
pfit <-coxph(Surv(time,status>0) ~ trt + log(bili) + log(protime) + age +
             platelet, data=pbcc)
plot(survfit(Surv(time, status>0) ~ trt, data=pbcc), mark.time=FALSE)
lines(survexp( ~ trt, ratetable=pfit, data=pbcc), col='purple')
```

survexp.fit

```
survexp.fit(group, x, y, times, death, ratetable)
```

group

x ratetable

y

times

death TRUEFALSEsurvexp

ratetable survexp.uswhite

y

timesytimes

timesy

survexp

survexpsurvexp.us

survexp.object

survexp
summaryprintplotpointslinessurvfit

surv
n.risk
time
std.err
strata time
method
na.action
call

survfit

n.risk

[plot.survfitsummary.survexpprint.survfitsurvexp](#)

survfit

survfit(formula, ...)

formula
...

?survfit.formula?survfit.coxph
dim(fit)coxph

survfit

survfit(Surv(time, status))

[survfit.formulasurvfit.coxphsurvfit.objectprint.survfitplot.survfit](#)
[quantile.survfitresiduals.survfitsummary.survfit](#)

survfit.coxph

```
## S3 method for class 'coxph'
survfit(formula, newdata,
        se.fit=TRUE, conf.int=.95, individual=FALSE, stype=2, ctype,
        conf.type=c("log","log-log","plain","none", "logit", "arcsin"),
        censor=TRUE, start.time, id, influence=FALSE,
        na.action=na.pass, type, time0=FALSE, ...)
## S3 method for class 'coxphms'
survfit(formula, newdata,
        se.fit=FALSE, conf.int=.95, individual=FALSE, stype=2, ctype,
        conf.type=c("log","log-log","plain","none", "logit", "arcsin"),
        censor=TRUE, start.time, id, influence=FALSE,
        na.action=na.pass, type, p0=NULL, time0= FALSE, ...)

formula      coxph
newdata      coxphnewdata
se.fit       TRUEFALSE
conf.int
individual   id
stype
ctype
conf.type    "none""plain""log""log-log""logit"curve +- k *se(curve)conf.int
censor
id           newdatanewdatacoxphnewdatanewidnewdata
start.time   start.timestart.time
influence
na.action
type         stypectype
p0
time0
...
```

```

coxphstype=1mstate
ctypes=2tiescoxphcoxphmsctype=1coxph
newdatameansmeans
coxphnewdata
newdataidSurv

```

survfit

"survfit"survfit.objectprintplotlinespoints

```

model=TRUEfit <- coxph(...); survfit(fit)model.frame
log[S(t; z)]zlog[S(t; x)] = e^{(x-z)\beta} log[S(t; z)]xz = 0zx - zxnewdata

```

print.survfitplot.survfitlines.survfitcoxphSurvstrata

survfit.formula

```

## S3 method for class 'formula'
survfit(formula, data, weights, subset, na.action,
         stype=1, ctype=1, id, cluster, robust, istate, timefix=TRUE,
         etype, model=FALSE, error, entry=FALSE, time0=FALSE, ...)

```

formula	Surv~strata~ 1
data	subsetweights
weights	subset
subset	
na.action	subsetoptions()\$na.action
stype	

```

ctype
id
cluster
robust
istate          time0 =TRUE
timefix         aeqSurv
etype
model
error
entry          n.enterid
time0
...            survfit
               TRUEpstatecumhazse.fit = FALSE
               "none""plain""log""log-log""logit""arcsin"curve +- k *se(curve)
               conf.int
               "usual""peto""modified"

               start.time

               start.time
               entry=TRUEn.enterid
               stypectypestype=1, ctype=1stype=2, ctype=1stype=2, ctype=2.

```

```

dataformulaweightssubsetidclusteristate
type
robust= FALSE
 $p(t_0)H(t_1)H(t_2)\dots p(t_0)t_0t_1,t_2,\dots$ 
idtime0 = TRUE

```

```

p0istate

```

```

time0 = TRUE
robustrobustclusterididcluster
influenceresidinfluence
 $U(t)$ rowsum(wU, cluster)
model.frame

```

```

"survfit"survfit.objectprintplotlinespointsresidual

```


[survfit.coxph](#)[survfit.object](#)[print.survfit](#)[plot.survfit](#)[lines.survfit](#)[residuals.survfit](#)[coxphSurv](#)

```
#fit a Kaplan-Meier and plot it
fit <- survfit(Surv(time, status) ~ x, data = aml)
plot(fit, lty = 2:3)
legend(100, .8, c("Maintained", "Nonmaintained"), lty = 2:3)

#fit a Cox proportional hazards model and plot the
#predicted survival for a 60 year old
fit <- coxph(Surv(futime, fustat) ~ age, data = ovarian)
plot(survfit(fit, newdata=data.frame(age=60)),
     xscale=365.25, xlab = "Years", ylab="Survival")

# Here is the data set from Turnbull
# There are no interval censored subjects, only left-censored (status=3),
# right-censored (status 0) and observed events (status 1)
#
#
#           Time
#           1   2   3   4
# Type of observation
#       death    12   6   2   3
#       losses    3   2   0   3
#       late entry 2   4   2   5
#
tdata <- data.frame(time =c(1,1,1,2,2,2,3,3,3,4,4,4),
                    status=rep(c(1,0,2),4),
                    n      =c(12,3,2,6,2,4,2,0,2,3,3,5))
fit <- survfit(Surv(time, time, status, type='interval') ~1,
               data=tdata, weight=n)

#
# Three curves for patients with monoclonal gammopathy.
# 1. KM of time to PCM, ignoring death (statistically incorrect)
# 2. Competing risk curves (also known as "cumulative incidence")
# 3. Multi-state, showing Pr(in each state, at time t)
#
fitKM <- survfit(Surv(stop, event=='pcm') ~1, data=mgus1,
                 subset=(start==0))
fitCR <- survfit(Surv(stop, event) ~1,
                 data=mgus1, subset=(start==0))
fitMS <- survfit(Surv(start, stop, event) ~ 1, id=id, data=mgus1)
```

```
## Not run:
# CR curves show the competing risks
plot(fitCR, xscale=365.25, xmax=7300, mark.time=FALSE,
      col=2:3, xlab="Years post diagnosis of MGUS",
      ylab="P(state)")
lines(fitKM, fun='event', xmax=7300, mark.time=FALSE,
      conf.int=FALSE)
text(3652, .4, "Competing risk: death", col=3)
text(5840, .15, "Competing risk: progression", col=2)
text(5480, .30, "KM:prog")

## End(Not run)
```

survfit.matrix

```
## S3 method for class 'matrix'
survfit(formula, p0, method = c("discrete", "matexp"),
        start.time, ...)
```

```
formula
p0          p0
method
start.time
...
```

```
method
```

[survfit](#)

```

etime <- with(mgus2, ifelse(pstat==0, futime, ptime))
event <- with(mgus2, ifelse(pstat==0, 2*death, 1))
event <- factor(event, 0:2, labels=c("censor", "pcm", "death"))

cfit1 <- coxph(Surv(etime, event=="pcm") ~ age + sex, mgus2)
cfit2 <- coxph(Surv(etime, event=="death") ~ age + sex, mgus2)

# predicted competing risk curves for a 72 year old with mspike of 1.2
# (median values), male and female.
# The survfit call is a bit faster without standard errors.
newdata <- expand.grid(sex=c("F", "M"), age=72, mspike=1.2)

AJmat <- matrix(list(), 3,3)
AJmat[1,2] <- list(survfit(cfit1, newdata, std.err=FALSE))
AJmat[1,3] <- list(survfit(cfit2, newdata, std.err=FALSE))
csurv <- survfit(AJmat, p0 =c(entry=1, PCM=0, death=0))

```

survfit.object

```

survfitc('survfitms', 'survfit')
printsummaryplotpointslinesprint.survfit

```

```

n
time
n.risk
n.event
n.enter      id
n.censor
surv
pstate      pstatesurv
std.err
cumhaz
counts      n.riskcounts
strata      time
upper
lower
t0
p0sp0      sp0
newdata
n.id
conf.type
conf.int

```

transitions
na.action
call
type
influence.pinfluence.c
pstatesurvcumhaz
version

survfitsurvfitms

survfit

survfit
t0p0
survfit0

[plot.survfitsummary.survfitprint.survfitsurvfitsurvfit0](#)

survfit0

survfit0(x, ...)

x
...

survfitt0
plot.survfitsummary.survfit

t0

survfit

timesurvstatecumhazstd.errstd.cumhaz

survfitcoxph.fit

```
survfitcoxph.fit(y, x, wt, x2, risk, newrisk, strata, se.fit, survtype,  
vartype, varmat, id, y2, strata2, unlist=TRUE)
```

y	
x	
wt	
x2	x
risk	
newrisk	
strata	
se.fit	TRUE
survtype	coxph
vartype	
varmat	
id	nrow(x2)x2x2
y2	x2
strata2	x2
unlist	FALSEsurvfit

survfit

survfit.coxphinst/sourcecode.pdf

[survfit.coxph](#)

survfit_confint

```
survfit_confint(p, se, logse = TRUE, conf.type, conf.int = 0.95, selow, ulimit = TRUE)
```

```
p          survpstatecumhaz
se
logse      se
conf.type
conf.int
selow
ulimit
```

lowerupper

[survfit.object](#)

survival-deprecated

```
survConcordance(formula, data, weights, subset, na.action) # use concordance
survConcordance.fit(y, x, strata, weight)    # use concordancefit
```

```
formula      ~Surv
data
weightssubsetna.action
              coxph
xystrataweight
```

[Deprecated](#)

survobrien

```
survobrien(formula, data, subset, na.action, transform)
```

formula

data formulasubsetweights

subset

na.action options()\\$na.action

transform

Surv.strata.I()

coxph

[survdiff](#)

```
xx <- survobrien(Surv(futime, fustat) ~ age + factor(rx) + I(ecog.ps),  
                 data=ovarian)  
coxph(Surv(time, status) ~ age + strata(.strata.), data=xx)
```

survreg

```
survreg(formula, data, weights, subset,  
        na.action, dist="weibull", init=NULL, scale=0,  
        control, parms=NULL, model=FALSE, x=FALSE,  
        y=TRUE, robust=FALSE, cluster, score=FALSE, ...)
```

formula	SurvSurvlmformula
data	formulaweightssubset
weights	
subset	
na.action	subsetoptions()\\$na.action
dist	survreg.distributions "weibull""exponential""gaussian" "logistic""lognormal""loglogistic" survreg.distributions
parms	
init	
scale	
control	survreg.control survreg.control()
modelxy	
score	
robust	cluster
cluster	modeldata
...	survreg.control

model.frame

survreg

[survreg.objects](#)[survreg.distributions](#)[pspline](#)[frailty](#)[ridge](#)


```

# Fit an exponential model: the two fits are the same
survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian, dist='weibull',
        scale=1)
survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian,
        dist="exponential")

#
# A model with different baseline survival shapes for two groups, i.e.,
# two different scale parameters
survreg(Surv(time, status) ~ ph.ecog + age + strata(sex), lung)

# There are multiple ways to parameterize a Weibull distribution. The survreg
# function embeds it in a general location-scale family, which is a
# different parameterization than the rweibull function, and often leads
# to confusion.
# survreg's scale = 1/(rweibull shape)
# survreg's intercept = log(rweibull scale)
# For the log-likelihood all parameterizations lead to the same value.
y <- rweibull(1000, shape=2, scale=5)
survreg(Surv(y)~1, dist="weibull")

# Economists fit a model called `tobit regression', which is a standard
# linear regression with Gaussian errors, and left censored data.
tobinfit <- survreg(Surv(durable, durable>0, type='left') ~ age + quant,
                   data=tobin, dist='gaussian')

```

survreg.control

[survreg](#)

```

survreg.control(maxiter=30, rel.tolerance=1e-09,
toler.chol=1e-10, iter.max, debug=0, outer.max=10)

```

```

maxiter
rel.tolerance
toler.chol
iter.max      maxiter
debug
outer.max

```

[survreg](#)

survreg.distributions

Ff

survreg.distributions

$$F1 - Fff'/ff''/f$$

extremegaussianlogistic

$$F = 1 - e^{-e^t}.$$

weibulllognormalloglogisticsurvregweibull

exponentialrayleighscaleloggaussianlognormal

survreg.distributionssurvreg.distributions

survregweibullpnormplogisptsurvregDtest

```

# time transformation
survreg(Surv(time, status) ~ ph.ecog + sex, dist='weibull', data=lung)
# change the transformation to work in years
# intercept changes by log(365), everything else stays the same
my.weibull <- survreg.distributions$weibull
my.weibull$trans <- function(y) log(y/365)
my.weibull$itrans <- function(y) 365*exp(y)
survreg(Surv(time, status) ~ ph.ecog + sex, lung, dist=my.weibull)

# Weibull parametrisation
y<-rweibull(1000, shape=2, scale=5)
survreg(Surv(y)~1, dist="weibull")
# survreg scale parameter maps to 1/shape, linear predictor to log(scale)

# Cauchy fit
mycauchy <- list(name='Cauchy',
  init= function(x, weights, ...)
    c(median(x), mad(x)),
  density= function(x, parms) {
    temp <- 1/(1 + x^2)
    cbind(.5 + atan(x)/pi, .5+ atan(-x)/pi,
      temp/pi, -2 *x*temp, 2*temp*(4*x^2*temp -1))
  },
  quantile= function(p, parms) tan((p-.5)*pi),
  deviance= function(...) stop('deviance residuals not defined')
)
survreg(Surv(log(time), status) ~ ph.ecog + sex, lung, dist=mycauchy)

```

survreg.object

survregprintsummarypredictresiduals

survreg

linear.predictors

linear.predictorsweightsxymodelcalltermsformulalm

[survreglm](#)

survregDtest

survreg

survregDtest(dlist, verbose = F)

dlist

verbose

survreg

[survreg.distributionssurvreg](#)

```
# An invalid distribution (it should have "init =" on line 2)
# survreg would give an error message
mycauchy <- list(name='Cauchy',
  init<- function(x, weights, ...)
    c(median(x), mad(x)),
  density= function(x, parms) {
    temp <- 1/(1 + x^2)
    cbind(.5 + atan(temp)/pi, .5+ atan(-temp)/pi,
      temp/pi, -2 *x*temp, 2*temp^2*(4*x^2*temp -1))
  },
  quantile= function(p, parms) tan((p-.5)*pi),
  deviance= function(...) stop('deviance residuals not defined')
)

survregDtest(mycauchy, TRUE)
```

survSplit

```
survSplit(formula, data, subset, na.action=na.pass,
          cut, start="tstart", id, zero=0, episode,
          end="tstop", event="event", added)
```

formula

data

subsetna.action

cut

start

id

zero start

episode

end

event

added

episodeadded

```
~ .Surv(adam, joe, fred)~.tstartendeventSurv(time, stat==2)
```

na.pass

Survcutreshape

```
fit1 <- coxph(Surv(time, status) ~ karno + age + trt, veteran)
plot(cox.zph(fit1)[1])
# a cox.zph plot of the data suggests that the effect of Karnofsky score
# begins to diminish by 60 days and has faded away by 120 days.
# Fit a model with separate coefficients for the three intervals.
#
vet2 <- survSplit(Surv(time, status) ~., veteran,
                  cut=c(60, 120), episode = "timegroup")
fit2 <- coxph(Surv(tstart, time, status) ~ karno* strata(timegroup) +
```

```

      age + trt, data= vet2)
c(overall= coef(fit1)[1],
  t0_60 = coef(fit2)[1],
  t60_120= sum(coef(fit2)[c(1,4)]),
  t120 = sum(coef(fit2)[c(1,5)]))

# Sometimes we want to split on one scale and analyse on another
# Add a "current age" variable to the mgus2 data set.
temp1 <- mgus2
temp1$endage <- mgus2$age + mgus2$futime/12 # futime is in months
temp1$startage <- temp1$age
temp2 <- survSplit(Surv(age, endage, death) ~ ., temp1, cut=25:100,
  start= "age1", end= "age2")

# restore the time since enrollment scale
temp2$time1 <- (temp2$age1 - temp2$startage)*12
temp2$time2 <- (temp2$age2 - temp2$startage)*12

# In this data set, initial age and current age have similar utility
mfit1 <- coxph(Surv(futime, death) ~ age + sex, data=mgus2)
mfit2 <- coxph(Surv(time1, time2, death) ~ age1 + sex, data=temp2)

```

tcut

```

tcut(x, breaks, labels, scale=1)
## S3 method for class 'tcut'
levels(x)

```

```

x
breaks
labels
scale          xbreaks

```

tcut

cutpyears

```

# For pyears, all time variable need to be on the same scale; but
# futime is in months and age is in years
test <- mgus2
test$years <- test$futime/30.5 # follow-up in years

# first grouping based on years from starting age (= current age)

```

```

# second based on years since enrollment (all start at 0)
test$agegrp <- tcut(test$age, c(0,60, 70, 80, 100),
                    c("<=60", "60-70", "70-80", ">80"))
test$fgrp <- tcut(rep(0, nrow(test)), c(0, 1, 5, 10, 100),
                  c("0-1yr", "1-5yr", "5-10yr", ">10yr"))

# death rates per 1000, by age group
pfit1 <- pyears(Surv(years, death) ~ agegrp, scale =1000, data=test)
round(pfit1$event/ pfit1$pyears)

#death rates per 100, by follow-up year and age
# there are excess deaths in the first year, within each age stratum
pfit2 <- pyears(Surv(years, death) ~ fgrp + agegrp, scale =1000, data=test)
round(pfit2$event/ pfit2$pyears)

```

timeline

```

totimeline(formula, data, id, istate)
fromtimeline(formula, data, id, istate="istate")

```

formula	Surv
data	
id	dataid
istate	totimelinefromtimeline

```

survalidid
tmerge
tmerge

```

[tmergesurvSplit](#)

tmerge

```
tmerge(data1, data2, id,..., tstart, tstop, options)
```

data1

data2

id

...

tstart

tstop

options

```
yxxxinittdcstartxna.rm=TRUEx
```

```
x
```

```
x
```

```
tstarttstopidoptionsididnamedata1tstart
```

```
na.rmdata2delay
```

```
tm.retaintcountdata2data1
```

```
tdceventeventtdceventcumtdccumevent
```

[neardate](#)


```

# The pbc data set contains baseline data and follow-up status
# for a set of subjects with primary biliary cirrhosis, while the
# pbcseq data set contains repeated laboratory values for those
# subjects.
# The first data set contains data on 312 subjects in a clinical trial plus
# 106 that agreed to be followed off protocol, the second data set has data
# only on the trial subjects.
temp <- subset(pbc, id <= 312, select=c(id:sex, stage)) # baseline data
pbc2 <- tmerge(temp, temp, id=id, endpt = event(time, status))
pbc2 <- tmerge(pbc2, pbcseq, id=id, ascites = tdc(day, ascites),
              bili = tdc(day, bili), albumin = tdc(day, albumin),
              protime = tdc(day, protime), alk.phos = tdc(day, alk.phos))

fit <- coxph(Surv(tstart, tstop, endpt==2) ~ protime + log(bili), data=pbc2)

```

tobin

```

tobin
data(tobin, package="survival")

```

```

tfit <- survreg(Surv(durable, durable>0, type='left') ~age + quant,
               data=tobin, dist='gaussian')

predict(tfit,type="response")

```

transplant

```
transplant
data(transplant, package="survival")
```

```
age
sex mf
abo ABABO
year
ftime
event censoreddeathltxwithdraw
```

transplanttransplant2

```
#since event is a factor, survfit creates competing risk curves
pfit <- survfit(Surv(ftime, event) ~ abo, transplant)
pfit[,2] #time to liver transplant, by blood type
plot(pfit[,2], mark.time=FALSE, col=1:4, lwd=2, xmax=735,
      xscale=30.5, xlab="Months", ylab="Fraction transplanted",
      xaxt = 'n')
temp <- c(0, 6, 12, 18, 24)
axis(1, temp*30.5, temp)
legend(450, .35, levels(transplant$abo), lty=1, col=1:4, lwd=2)

# competing risks for type 0
plot(pfit[4,], xscale=30.5, xmax=735, col=1:3, lwd=2)
legend(450, .4, c("Death", "Transplant", "Withdrawal"), col=1:3, lwd=2)
```

udca

```
udca
udca2
data(udca, package="survival")
```

```
id
trt
entry.dt
last.dt
stage
bili
riskscore
death.dt
tx.dt
hprogress.dt
varices.dt
ascites.dt
enceph.dt
double.dt
worsen.dt
```

```
udca1udca2
```

```
# values found in table 8.3 of the book
fit1 <- coxph(Surv(futime, status) ~ trt + log(bili) + stage,
              cluster =id , data=udca1)
fit2 <- coxph(Surv(futime, status) ~ trt + log(bili) + stage +
              strata(endpoint), cluster=id, data=udca2)
```

untangle.specials

termsterms

untangle.specials(tt, special, order=1)

tt terms
special
order

vars
terms

```
formula <- Surv(tt,ss) ~ x + z*strata(id)
tms <- terms(formula, specials="strata")
## the specials attribute
attr(tms, "specials")
## main effects
untangle.specials(tms, "strata")
## and interactions
untangle.specials(tms, "strata", order=1:2)
```

uspop2

uspop

```
us50 <- uspop2[51:101,, "2000"] #US 2000 population, 50 and over
age <- as.integer(dimnames(us50)[[1]])
smat <- model.matrix( ~ factor(floor(age/5)) -1)
ustot <- t(smat) %*% us50 #totals by 5 year age groups
temp <- c(50,55, 60, 65, 70, 75, 80, 85, 90, 95)
dimnames(ustot) <- list(c(paste(temp, temp+4, sep="-"), "100+"),
                        c("male", "female"))
```

vcov.coxph

```
## S3 method for class 'coxph'
vcov(object, complete=TRUE, ...)
## S3 method for class 'coxphms'
vcov(object, complete=TRUE, matrix=FALSE, ...)
## S3 method for class 'survreg'
vcov(object, complete=TRUE, ...)

object
complete      coef(object)complete=TRUEcomplete
matrix        vcov(object, matrix=TRUE)[,i]coef(object, matrix=TRUE)[,i]
...

coxphsurvreg
```

veteran

```
veteran
data(cancer, package="survival")
```

xtfrm.Surv

```
## S3 method for class 'Surv'
xtfrm(x)
```

```
x          Surv
```

```
(tstart, tstop, status)
xtfrmordersort
```

```
x
```

[sortorder](#)

```
test <- c(Surv(c(10, 9,9, 8,8,8,7,5,5,4), rep(1:0, 5)), Surv(6.2, NA))
test
sort(test)
```

yates

```
yates(fit, term, population = c("data", "factorial", "sas"),
levels, test = c("global", "trend", "pairwise"), predict = "linear",
options, nsim = 200, method = c("direct", "sgtt"))
```

```
fit
term
population
levels      term
test
predict
options
nsim
method
```

yates
estimate
tests
mvar
summary

```
fit1 <- lm(skips ~ Solder*Opening + Mask, data = solder)
yates(fit1, ~Opening, population = "factorial")

fit2 <- coxph(Surv(time, status) ~ factor(ph.ecog)*sex + age, lung)
yates(fit2, ~ ph.ecog, predict="risk") # hazard ratio
```

yates_setup	yates
-------------	-------

yates

yates_setup(fit, ...)

fit
...

yates

yates