Package ‘gtools’
March 24, 2013

Title Various R programming tools
Description Various R programming tools
Version 2.7.1
Date 2013-03-16
Author Gregory R. Warnes. Includes R source code and/or documentation contributed by Ben Bolker and Thomas Lumley
Maintainer Gregory R. Warnes <greg@warnes.net>
License LGPL-2.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2013-03-24 07:58:20

R topics documented:

addLast .................................................. 2
ask .......................................................... 3
binsearch .................................................. 4
capture .................................................... 7
checkRVersion .......................................... 8
combinations ............................................. 9
defmacro .................................................. 10
ELISA .................................................. 13
foldchange .............................................. 14
gtools-deprecated .................................... 15
invalid ..................................................... 15
keywords ............................................... 16
logit ..................................................... 17
mixedsort ............................................... 18
odd ......................................................... 20
addLast

Description
Add a function to be executed when R exits.

Usage
addLast(fun)

Arguments
fun Function to be called.

Details
addLast defines .Last (if not already present) or redefines it so that the function fun will be called when R exits. The latter is accomplished by saving the current definition of .Last and creating a new .Last function that calls fun and then the original .Last function.

Value
None.

Author(s)
Gregory R. Warnes <greg@warnes.net>

See Also
.Last
Examples

## Not run:
## Print a couple of cute messages when R exits.
helloWorld <- function() cat("\nHello World!\n")
byeWorld <- function() cat("\nGoodbye World!\n")

addLast(byeWorld)
addLast(helloWorld)

q("no")

## Should yield:
##
## Save workspace image? [y/n/c]: n
## Hello World!
## Goodbye World!
## Process R finished at Tue Nov 22 10:28:55 2005

## Unix-flavour example: send Rplots.ps to printer on exit.
myLast <- function()
{
  cat("Now sending PostScript graphics to the printer:\n")
  system("lpr Rplots.ps")
  cat("bye bye...\n")
}
addLast(myLast)
quit("yes")

## Should yield:
##
## Now sending PostScript graphics to the printer:
## lpr: job 1341 queued
## bye bye...
## Process R finished at Tue Nov 22 10:28:55 2005

## End(Not run)

---

ask

Display a prompt and collect the user’s response

Description

Display a prompt and collect the user’s response
Usage

ask(msg = "Press <RETURN> to continue: ")

Arguments

msg Character vector providing the message to be displayed

Details

The prompt message will be displayed, and then `readLines` is used to collect a single input value (possibly empty), which is then returned.

Value

A character scalar containing the input provided by the user.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

`readLines`, `scan`

Examples

```r
# use default prompt
ask()

silly <- function()
{
  age <- ask("How old are you? ")
  age <- as.numeric(age)
  cat("In 10 years you will be", age+10, "years old!\n")
}
```

binsearch  Binary Search

Description

Search within a specified range to locate an integer parameter which results in the specified monotonic function obtaining a given value.
Usage

binsearch(fun, range, ..., target = 0, lower = ceiling(min(range)),
upper = floor(max(range)), maxiter = 100, showiter = FALSE)

Arguments

dfun Monotonic function over which the search will be performed.
range 2-element vector giving the range for the search.
... Additional parameters to the function fun.
target Target value for fun. Defaults to 0.
lower Lower limit of search range. Defaults to min(range).
upper Upper limit of search range. Defaults to max(range).
maxiter Maximum number of search iterations. Defaults to 100.
showiter Boolean flag indicating whether the algorithm state should be printed at each
iteration. Defaults to FALSE.

Details

This function implements an extension to the standard binary search algorithm for searching a sorted list. The algorithm has been extended to cope with cases where an exact match is not possible, to detect whether the function may be monotonic increasing or decreasing and act appropriately, and to detect when the target value is outside the specified range.

The algorithm initializes two variable lo and hi to the extremes values of range. It then generates a new value center halfway between lo and hi. If the value of fun at center exceeds target, it becomes the new value for lo; otherwise it becomes the new value for hi. This process is iterated until lo and hi are adjacent. If the function at one or the other equals the target, this value is returned, otherwise lo, hi, and the function value at both are returned.

Note that when the specified target value falls between integers, the two closest values are returned. If the specified target falls outside of the specified range, the closest endpoint of the range will be returned, and an warning message will be generated. If the maximum number if iterations was reached, the endpoints of the current subset of the range under consideration will be returned.

Value

A list containing:

call How the function was called.
numiter The number of iterations performed
flag One of the strings, "Found", "Between Elements", "Maximum number of iterations reached", "Reached lower boundary", or "Reached upper boundary."
where One or two values indicating where the search terminated.
value Value of the function fun at the values of where.
Note

This function often returns two values for `where` and `value`. Be sure to check the flag parameter to see what these values mean.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

`optim`, `optimize`, `uniroot`

Examples

### Toy examples

# search for x=10
binsearch( function(x) x-10, range=c(0,20) )

# search for x=10.1
binsearch( function(x) x-10.1, range=c(0,20) )

### Classical toy example

# binary search for the index of 'M' among the sorted letters
fun <- function(X) ifelse(LETTERS[X] > 'M', 1,
  ifelse(LETTERS[X] < 'M', -1, 0 )
)

binsearch( fun, range=1:26 )
  # returns $where=13
LETTERS[13]

### Substantive example, from genetics

## Not run:
library(genetics)
# Determine the necessary sample size to detect all alleles with
# frequency 0.07 or greater with probability 0.95.
power.fun <- function(N) 1 - gregorius(N=N, freq=0.07)$missprob

binsearch( power.fun, range=c(0,100), target=0.95 )

  # equivalent to
  gregorius( freq=0.07, missprob=0.05)

## End(Not run)
capture

Capture printed output of an R expression in a string

Description

Capture printed output of an R expression in a string

Usage

capture(expression, collapse = "\n")
sprint(x,...)

Arguments

expression  R expression whose output will be captured.
collapse  Character used to join output lines. Defaults to "\n". Use NULL to return a vector of individual output lines.
x  Object to be printed
...  Optional parameters to be passed to print

Details

The capture function uses sink to capture the printed results generated by expression.
The function sprint uses capture to redirect the results of calling print on an object to a string.

Value

A character string, or if collapse==NULL a vector of character strings containing the printed output from the R expression.

WARNING

R 1.7.0+ includes capture.output, which duplicates the functionality of capture. Thus, capture is depreciated.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

texteval, capture.output
Examples

# capture the results of a loop
loop.text <- capture( for(i in 1:10) cat("i=",i,"\n") )
loop.text

# put regression summary results into a string
data(iris)
reg <- lm( Sepal.Length ~ Species, data=iris )
summary.text <- sprint( summary(reg) )
cat(summary.text)

checkRVersion

Check if a newer version of R is available

Description
Check if a newer version of R is available

Usage

checkRVersion(quiet = FALSE)

Arguments

quiet Logical indicating whether printed output should be supressed.

Details
This function accesses the R web site to discover the latest released version of R. It then compares
this version to the running version. If the running version is the same as the latest version, it prints
the message, "The latest version of R is installed:" followed by the version number, and returns
NULL. If the running version is older than the current version, it displays the message, "A newer
version of R is now available:" followed by the corresponding version number, and returns the
version number.
If quiet=TRUE, no printing is performed.

Value
Either the version number of the latest version of R, if the running version is less than the latest
version, or NULL.

Note
This function utilizes the internet to access the R project web site. If internet access is unavailable,
the function will fail.
combinations

Author(s)
Gregory R. Warnes <gregory.warnes@rochester.edu>

See Also
R.Version

Examples

checkRVersion()

ver <- checkRVersion()
print(ver)

combinations

Enumerate the Combinations or Permutations of the Elements of a Vector

description

combinations enumerates the possible combinations of a specified size from the elements of a vector. permutations enumerates the possible permutations.

Usage

combinations(n, r, v=1:n, set=TRUE, repeats.allowed=FALSE)
permutations(n, r, v=1:n, set=TRUE, repeats.allowed=FALSE)

Arguments

n Size of the source vector
r Size of the target vectors
v Source vector. Defaults to 1:n
set Logical flag indicating whether duplicates should be removed from the source vector v. Defaults to TRUE.
repeats.allowed Logical flag indicating whether the constructed vectors may include duplicated values. Defaults to FALSE.

Details

Caution: The number of combinations and permutations increases rapidly with n and r!.
To use values of n above about 45, you will need to increase R’s recursion limit. See the expression argument to the options command for details on how to do this.
Value

Returns a matrix where each row contains a vector of length \( r \).

Author(s)

Original versions by Bill Venables &lt;Bill.Venables@cmis.csiro.au&gt;. Extended to handle repeats.allowed by Gregory R. Warnes &lt;greg@warnes.net&gt;.

References


See Also

choose, options

Examples

combinations(3,2,letters[1:3])
combinations(3,2,letters[1:3],repeats=TRUE)

permutations(3,2,letters[1:3])
permutations(3,2,letters[1:3],repeats=TRUE)

# To use large 'n', you need to change the default recusion limit
options(expressions=1e5)
cmat <- combinations(300,2)
dim(cmat) # 44850 by 2

defmacro          Define a macro

defmacro define a macro that uses R expression replacement
strmacro define a macro that uses string replacement

Usage

defmacro(..., expr)
strmacro(..., expr, strexpr)

Arguments

...                  macro argument list
expr                 R expression defining the macro body
strexpr              character string defining the macro body
Details

defmacro and strmacro create a macro from the expression given in expr, with formal arguments given by the other elements of the argument list.

A macro is similar to a function definition except for handling of formal arguments. In a function, formal arguments are simply variables that contains the result of evaluating the expressions provided to the function call. In contrast, macros actually modify the macro body by replacing each formal argument by the expression (defmacro) or string (strmacro) provided to the macro call.

For defmacro, the special argument name DOTS will be replaced by ... in the formal argument list of the macro so that ... in the body of the expression can be used to obtain any additional arguments passed to the macro. For strmacro you can mimic this behavior providing a DOTS="" argument. This is illustrated by the last example below.

Macros are often useful for creating new functions during code execution.

Value

A macro function.

Note

Note that because [the defmacro code] works on the parsed expression, not on a text string, defmacro avoids some of the problems of traditional string substitution macros such as strmacro and the C preprocessor macros. For example, in

```r
mul <- defmacro(a, b, expr={a*b})
```

a C programmer might expect `mul(i, j + k)` to expand (incorrectly) to `i*j + k`. In fact it expands correctly, to the equivalent of `i*(j + k)`. For a discussion of the differences between functions and macros, please Thomas Lumley’s R-News article (reference below).

Author(s)

Thomas Lumley wrote defmacro. Gregory R. Warnes <greg@warnes.net> enhanced it and created strmacro.

References

The original defmacro code was directly taken from:


See Also

function substitute, eval, parse, source, parse,
Examples

####
# macro for replacing a specified missing value indicator with NA
# within a dataframe
####
setNA <- defmacro(df, var, values, 
  expr={
    df$var[df$var %in% values] <- NA
  })

# create example data using 999 as a missing value indicator
d <- data.frame(  
  V1=c(1, 2, 3, 4, 5, 6, 999, 8, 9, 10),  
  V2=c(1, 1, 1, 1, 1, 2, 999, 2, 999, 999)  
)
d

# Try it out
setNA(d, V1, 999)
setNA(d, V2, 999)
d

####
# Expression macro
####
plot.d <- defmacro( df, var, DOTS, col="red", title="", expr=  
  plot( df$var ~ df$Grp, type="b", col=col, main=title, ... )  
)
plot.d( d, V1)
plot.d( d, V1, col="blue" )
plot.d( d, V1, lwd=4) # use optional 'DOTS' argument

####
# String macro (note the quoted text in the calls below)
#
# This style of macro can be useful when you are reading
# function arguments from a text file
####
plot.s <- strmacro( DF, VAR, COL="'red'", TITLE="'", DOTS="", expr=  
  plot( DF$VAR ~ DF$Grp, type="b", col=COL, main=TITLE, DOTS)  
)
plot.s( "d", "V1")
plot.s( DF="d", VAR="V1", COL="'blue'" )
plot.s( "d", "V1", DOTS="lwd=4") # use optional 'DOTS' argument

**********
# Create a macro that defines new functions

```r
plot.sf <- defmacro(type='b', col='black',
                   title=deparse(substitute(x)), DOTS, expr=
                   function(x,y) plot( x,y, type=type, col=col, main=title, ...)
)
```

```r
plot.red <- plot.sf(col='red',title='Red is more Fun!')
plot.blue <- plot.sf(col='blue',title="Blue is Best!", lty=2)
```

```r
plot.red(1:100,rnorm(100))
plot.blue(1:100,rnorm(100))
```

---

**ELISA**

*Data from an ELISA assay*

**Description**

Observed signals and (for some observations) nominal concentrations for samples that were aliquoted to multiple assay plates, which were read multiple times on multiple days.

**Usage**

```r
data(ELISA)
```

**Format**

A data frame with the following columns:

- **PlateDayfactor.** Specifies one of four physically distinct 96 well plates
- **Readfactor.** The signal was read 3 times for each plate.
- **Descriptioncharacter.** Indicates contents of sample.
- **Concentrationnumeric.** Nominal concentration of standards (NA for all other samples).
- **Signalnumeric.** Assay signal. Specifically, optical density (a colorimetric assay).

**Source**

Anonymized data.
foldchange

Description

foldchange computes the fold change for two sets of values. logratio2foldchange converts values from log-ratios to fold changes. foldchange2logratio does the reverse.

Usage

code

foldchange(num, denom)
logratio2foldchange(logratio, base=2)
foldchange2logratio(foldchange, base=2)

Arguments

num, denom vector/matrix of numeric values
logratio vector/matrix of log-ratio values
foldchange vector/matrix of fold-change values
base Exponential base for the log-ratio.

Details

Fold changes are commonly used in the biological sciences as a mechanism for comparing the relative size of two measurements. They are computed as: \( \frac{\text{num}}{\text{denom}} \) if \( \text{num} > \text{denom} \), and as \( -\frac{\text{denom}}{\text{num}} \) otherwise.

Fold-changes have the advantage of ease of interpretation and symmetry about \( \text{num} = \text{denom} \), but suffer from a discontinuity between -1 and 1, which can cause significant problems when performing data analysis. Consequently statisticians prefer to use log-ratios.

Value

A vector or matrix of the same dimensions as the input containing the converted values.

Author(s)

Gregory R. Warnes <greg@warnes.net>

Examples

```r
a <- 1:21
b <- 21:1
f <- foldchange(a, b)
cbind(a, b, f)
```
Description

These functions are provided for compatibility with older versions of gtools, and may be defunct as soon as the next release.

Usage

assert(FLAG)

Arguments

FLAG Expression that should evaluate to a boolean vector

Details

The original help page for these functions is often available at help("oldName-deprecated") (note the quotes).

assert is a deprecated synonym for stopifnot.

See Also

Deprecated

invalid Test if a value is missing, empty, or contains only NA or NULL values

Description

Test if a value is missing, empty, or contains only NA or NULL values.

Usage

invalid(x)

Arguments

x value to be tested

Value

Logical value.
Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

missing, is.na, is.null

Examples

invalid(NA)
invalid()
invalid(c(NA, NA, NULL, NA))

invalid(list(a=1, b=NULL))

# example use in a function
myplot <- function(x, y) {
  if(invalid(y)) {
    y <- x
    x <- 1:length(y)
  }
  plot(x, y)
}
myplot(1:10)
myplot(1:10, NA)

Description

List valid keywords for R man pages

Usage

keywords(...)

Arguments

... Optional arguments to pass to show.file()

Details

This function simply determines the path \$RHOME/doc/KEYWORDS and calls show.file() to display it.
Description

Compute generalized logit and generalized inverse logit functions.

Usage

logit(x, min = 0, max = 1)
inv.logit(x, min = 0, max = 1)

Arguments

x value(s) to be transformed
min Lower end of logit interval
max Upper end of logit interval

Details

The generalized logit function takes values on [min, max] and transforms them to span [-Inf,Inf] it is defined as:

\[ y = \log\left(\frac{p}{1 - p}\right) \]

where

\[ p = \frac{(x - \text{min})}{(\text{max} - \text{min})} \]

The generalized inverse logit function provides the inverse transformation:
mixedsort

\[ x = p'(\text{max} - \text{min}) + \text{min} \]

where

\[ p' = \frac{\exp(y)}{(1 + \exp(y))} \]

Value

Transformed value(s).

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

logit, inv.glogit

Examples

```r
x <- seq(0, 10, by=0.25)
x <- logit(x, min=0, max=10)
cbind(x, xt)
y <- inv.logit(xt, min=0, max=10)
cbind(x, xt, y)
```

---

mixedsort

Order or Sort strings with embedded numbers so that the numbers are in the correct order

Description

These functions sort or order character strings containing numbers so that the numbers are numerically sorted rather than sorted by character value. I.e. "Asprin 50mg" will come before "Asprin 100mg". In addition, case of character strings is ignored so that "a", will come before "B" and "C".

Usage

mixedsort(x)

Arguments

\[ x \]
Character vector to be sorted
mixedsort

Details

I often have character vectors (e.g. factor labels) that contain both text and numeric data, such as
compound and dose. This function is useful for sorting these character vectors into a logical order.

It does so by splitting each character vector into a sequence of character and numeric sections, and
then sorting along these sections, with numbers being sorted by numeric value (e.g. "50" comes
before "100"), followed by characters strings sorted by character value (e.g. "A" comes before
"B").

Empty strings are always sorted to the front of the list, and NA values to the end.

Value

mixedorder returns a vector giving the sort order of the input elements. mixedsort returns the
sorted vector.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

sort, order

Examples

# compound & dose labels
Treatment <- c("Control", "Asprin 10mg/day", "Asprin 50mg/day",
               "Asprin 100mg/day", "Acetomycin 100mg/day",
               "Acetomycin 1000mg/day")

# ordinary sort puts the dosages in the wrong order
sort(Treatment)
# but mixedsort does the 'right' thing
mixedsort(Treatment)

# Here is a more complex example
x <- rev(c("AA 0.50 ml", "AA 1.5 ml", "AA 500 ml", "AA 1500 ml",
           "EXP 1", "AA 1e3 ml", "A A A", "1 2 3 A", "NA", NA, "1e2",
           ",", ",-", "1A", "1 A", "100", "100A", "Inf"))

mixedorder(x)
mixedsort(x)
# notice that plain numbers, including 'Inf' show up before strings.
odd

Detect odd/even integers

Description

detect odd/even integers

Usage

odd(x)
even(x)

Arguments

x vector of integers

Value

Vector of TRUE/FALSE values.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

round

Examples

odd(4)
even(4)

odd(1:10)
even(1:10)
permute

Randomly Permute the Elements of a Vector

Description

Randomly Permute the elements of a vector

Usage

permute(x)

Arguments

x  Vector of items to be permuted

Details

This is simply a wrapper function for sample.

Value

Vector with the original items reordered.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

sample

Examples

x <- 1:10
permute(x)
quantcut

Create a Factor Variable Using the Quantiles of a Continuous Variable

Description
Create a factor variable using the quantiles of a continuous variable.

Usage
quantcut(x, q=seq(0, 1, by=0.25), na.rm=TRUE, ...)

Arguments
x  Continous variable.
q  Vector of quantiles used for creating groups. Defaults to seq(0, 1, by=0.25).
   See quantile for details.
na.rm  Boolean indicating whether missing values should be removed when computing
        quantiles. Defaults to TRUE.
...  Optional arguments passed to cut.

Details
This function uses quantile to obtain the specified quantiles of x, then calls cut to create a factor
variable using the intervals specified by these quantiles.

It properly handles cases where more than one quantile obtains the same value, as in the second
dexample below. Note that in this case, there will be fewer generated factor levels than the specified
number of quantile intervals.

Value
Factor variable with one level for each quantile interval given by q.

Author(s)
Gregory R. Warnes <greg@warnes.net>

See Also
cut, quantile
rdirichlet

Functions for the Dirichlet Distribution

Description
Functions to compute the density of or generate random deviates from the Dirichlet distribution.

Usage
rdirichlet(n, alpha)
ddirichlet(x, alpha)

Arguments
x A vector containing a single random deviate or matrix containing one random deviate per row.
n Number of random vectors to generate.
alpha Vector or (for ddirichlet) matrix containing shape parameters.

Details
The Dirichlet distribution is the multidimensional generalization of the beta distribution. It is the canonical Bayesian distribution for the parameter estimates of a multinomial distribution.
running

Value
ddirichlet returns a vector containing the Dirichlet density for the corresponding rows of x.
rdirichlet returns a matrix with n rows, each containing a single Dirichlet random deviate.

Author(s)
Code original posted by Ben Bolker to R-News on Fri Dec 15 2000. See http://www.r-project.org/nocvs/mail/r-help/2000/3865.html. Ben attributed the code to Ian Wilson <i.wilson@maths.abdn.ac.uk>. Subsequent modifications by Gregory R. Warnes <greg@warnes.net>.

See Also
dbeta, rbeta

Examples

```r
x <- rdirichlet(20, c(1,1,1) )
ddirichlet(x, c(1,1,1) )
```

Description
Applies a function over subsets of the vector(s) formed by taking a fixed number of previous points.

Usage

```r
running(X, Y=NULL, fun=mean, width=min(length(X), 20),
      allow.fewer=FALSE, pad=FALSE, align=c("right", "center","left"),
      simplify=TRUE, by, ...)
```

Arguments

- `X` : data vector
- `Y` : data vector (optional)
- `fun` : Function to apply. Default is mean
- `width` : Integer giving the number of vector elements to include in the subsets. Defaults to the lesser of the length of the data and 20 elements.
- `allow.fewer` : Boolean indicating whether the function should be computed for subsets with fewer than width points
running

    pad  Boolean indicating whether the returned results should be 'padded' with NAs corresponding to sets with less than width elements. This only applies when when allow.fewer is FALSE.
    align One of "right", "center", or "left". This controls the relative location of 'short' subsets with less then width elements: "right" allows short subsets only at the beginning of the sequence so that all of the complete subsets are at the end of the sequence (i.e. 'right aligned'), "left" allows short subsets only at the end of the data so that the complete subsets are 'left aligned', and "center" allows short subsets at both ends of the data so that complete subsets are 'centered'.
    simplify Boolean. If FALSE the returned object will be a list containing one element per evaluation. If TRUE, the returned object will be coerced into a vector (if the computation returns a scalar) or a matrix (if the computation returns multiple values). Defaults to FALSE.
    by Integer separation between groups. If by=width will give non-overlapping windows. Default is missing, in which case groups will start at each value in the X/Y range.

... parameters to be passed to fun

Details

    running applies the specified function to a sequential windows on X and (optionally) Y. If Y is specified the function must be bivariate.

Value

    List (if simplify==TRUE), vector, or matrix containing the results of applying the function fun to the subsets of X (running) or X and Y.

    Note that this function will create a vector or matrix even for objects which are not simplified by sapply.

Author(s)

    Gregory R. Warnes <greg@warnes.net>, with contributions by Nitin Jain <nitin.jain@pfizer.com>.

See Also

    wapply to apply a function over an x-y window centered at each x point, sapply, lapply

Examples

    # show effect of pad
    running(1:20, width=5)
    running(1:20, width=5, pad=TRUE)

    # show effect of align
    running(1:20, width=5, align="left", pad=TRUE)
    running(1:20, width=5, align="center", pad=TRUE)
    running(1:20, width=5, align="right", pad=TRUE)
# show effect of simplify
running(1:20, width=5, fun=function(x) x) # matrix
running(1:20, width=5, fun=function(x), simplify=FALSE) # list

# show effect of by
running(1:20, width=5) # normal
running(1:20, width=5, by=5) # non-overlapping
running(1:20, width=5, by=2) # starting every 2nd

# Use 'pad' to ensure correct length of vector, also show the effect
# of allow.fewer.
par(mfrow=c(2,1))
plot(1:20, running(1:20, width=5, allow.fewer=FALSE, pad=TRUE), type="b")
plot(1:20, running(1:20, width=5, allow.fewer=TRUE, pad=TRUE), type="b")
par(mfrow=c(1,1))

# plot running mean and central 2 standard deviation range
# estimated by *last* 40 observations
dat <- rnorm(500, sd=1 + (1:500)/500)
plot(dat)
sdfun <- function(x,sign=1) mean(x) + sign * sqrt(var(x))
lines(running(dat, width=51, pad=TRUE, fun=mean), col="blue")
lines(running(dat, width=51, pad=TRUE, fun=sdfun, sign=-1), col="red")
lines(running(dat, width=51, pad=TRUE, fun=sdfun, sign=1), col="red")

# plot running correlation estimated by last 40 observations (red)
# against the true local correlation (blue)
sd.Y <- seq(0,1,length=500)
X <- rnorm(500, sd=1)
Y <- rnorm(500, sd=sd.Y)
plot(running(X,X+Y,width=20,fun=cor,pad=TRUE),col="red",type="s")
r <- 1 / sqrt(1 + sd.Y^2) # true cor of (X,X+Y)
lines(r,type="l",col="blue")

---

scat  

Display debugging text

Description

IfgetOption('DEBUG')==TRUE, write text to STDOUT and flush so that the text is immediately displayed. Otherwise, do nothing.

Usage

scat(...)
setTCPNoDelay

Arguments

... Arguments passed to cat

Value

NULL (invisibly)

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

cat

Examples

options(DEBUG=NULL) # make sure DEBUG isn't set
scat("Not displayed")

options(DEBUG=TRUE)
scat("This will be displayed immediately (even in R BATCH output \n")
scat("files), provided options()$DEBUG is TRUE.")

setTCPNoDelay

Modify the TCP\_NODELAY ('de-Nagle') flag for socket objects

Description

Modify the TCP\_NODELAY ('de-Nagle') flag for socket objects

Usage

setTCPNoDelay(socket, value=TRUE)

Arguments

socket A socket connection object
value Logical indicating whether to set (TRUE) or unset (FALSE) the flag

Details

By default, TCP connections wait a small fixed interval before actually sending data, in order to permit small packets to be combined. This algorithm is named after its inventor, John Nagle, and is often referred to as 'Nagling'.

While this reduces network resource utilization in these situations, it imposes a delay on all outgoing message data, which can cause problems in client/server situations.

This function allows this feature to be disabled (de-Nagling, value=TRUE) or enabled (Nagling, value=FALSE) for the specified socket.
smartbind

Value

The character string "SUCCESS" will be returned invisible if the operation was successful. On failure, an error will be generated.

Author(s)

Gregory R. Warnes <greg@warnes.net>

References

"Nagle's algorithm" at WhatIS.com http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci754347,00.html


See Also

make.socket, socketConnection

Examples

## Not run:
  s <- make.socket(host='www.r-project.org', port=80)
  setTCPNoDelay(s, value=TRUE)

## End(Not run)

---

smartbind  Efficient rbind of data frames, even if the column names don't match

Description

Efficient rbind of data frames, even if the column names don’t match

Usage

smartbind(..., fill=NA, sep=':', verbose=FALSE)

Arguments

... Data frames to combine
fill Value to use when 'filling' missing columns. Defaults to NA.
sep Character string used to separate column names when pasting them together.
verbose Logical flag indicating whether to display processing messages. Defaults to FALSE.
Value

The returned data frame will contain:

- **columns**: all columns present in any provided data frame
- **rows**: a set of rows from each provided data frame, with values in columns not present in the given data frame filled with missing (NA) values.

The data type of columns will be preserved, as long as all data frames with a given column name agree on the data type of that column. If the data frames disagree, the column will be converted into a character strings. The user will need to coerce such character columns into an appropriate type.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

rbind, cbind

Examples

```r
df1 <- data.frame(A=1:10, B=LETTERS[1:10], C=rnorm(10) )
df2 <- data.frame(A=11:20, D=rnorm(10), E=letters[1:10] )

# rbind would fail
## Not run:
rbind(df1, df2)

# Error in match.names(clabs, names(xi)) : names do not match previous
# names:
# D, E

## End(Not run)
# but smartbind combines them, appropriately creating NA entries
smartbind(df1, df2)

# specify fill=0 to put 0 into the missing row entries
smartbind(df1, df2, fill=0)
```
Index

*Topic **IO**
  ask, 3
  capture, 7
*Topic **arith**
  odd, 20
*Topic **datasets**
  ELISA, 13
*Topic **distribution**
  permute, 21
  rdirichlet, 23
*Topic **documentation**
  keywords, 16
*Topic **manip**
  combinations, 9
  mixedsort, 18
  quantcut, 22
  smartbind, 28
*Topic **math**
  foldchange, 14
  logit, 17
*Topic **misc**
  gtools-deprecated, 15
  running, 24
  setTCPNoDelay, 27
*Topic **optimize**
  binsearch, 4
*Topic **print**
  capture, 7
  scat, 26
*Topic **programming**
  addLast, 2
  binsearch, 4
  defmacro, 10
  invalid, 15
  setTCPNoDelay, 27
*Topic **univar**
  mixedsort, 18
*Topic **utilities**
  checkRVersion, 8

  setTCPNoDelay, 27
. Last, 2
addLast, 2
ask, 3
assert (gtools-deprecated), 15
binsearch, 4
capture, 7
capture.output, 7
cat, 27
cbind, 29
checkRVersion, 8
choose, 10
combinations, 9
cut, 22
dbeta, 24
ddirichlet(rdirichlet), 23
defmacro, 10
Deprecated, 15
ELISA, 13
eval, 11
even (odd), 20
foldchange, 14
foldchange2logratio (foldchange), 14
function, 11
gtools-deprecated, 15
help, 17
inv.glogit, 18
inv.logit (logit), 17
invalid, 15
is.na, 16
is.null, 16
keywords, 16
lapply, 25
logit, 17, 18
logratio2foldchange (foldchange), 14

make.socket, 28
missing, 16
mixedorder (mixedsort), 18
mixedsort, 18

odd, 20
optim, 6
optimize, 6
options, 10
order, 19

parse, 11
permutations (combinations), 9
permute, 21
print, 7

quantcut, 22
quantile, 22

R.Version, 9
rbeta, 24
rbind, 29
rdirichlet, 23
readLines, 4
round, 20
running, 24

sample, 21
sapply, 25
scan, 4
scat, 26
setTCPNoDelay, 27
sink, 7
smartbind, 28
socketConnection, 28
sort, 19
source, 11
sprint (capture), 7
stopifnot, 15
strmacro (defmacro), 10
substitute, 11

texteval, 7

unirroot, 6

wapply, 25