Package ‘broman’

May 28, 2017

Version 0.65-4
Date 2017-05-07
Title Karl Broman's R Code
Description Miscellaneous R functions, including functions related to
    graphics (mostly for base graphics), permutation tests, running
    mean/median, and general utilities.
Author Karl W Broman <kbroman@biostat.wisc.edu>, with contributions from Aimee
    Teo Broman
Maintainer Karl W Broman <kbroman@biostat.wisc.edu>
Depends R (>= 2.15.0)
Imports utils, graphics, grDevices, stats, assertthat, RPushbullet,
    jsonlite, ggplot2, grid
Suggests testthat, devtools, roxygen2
License GPL-3
URL https://github.com/kbroman/broman
LazyData true
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-05-28 19:08:03 UTC

R topics documented:

  add_commas .................................................. 3
  arrowlocator ................................................ 3
  attrnames ..................................................... 4
  brocolors ...................................................... 5
  bromanversion ............................................... 6
  cf ............................................................ 7
  chisq ........................................................ 8
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>colwalpha ........................................... 9</td>
</tr>
<tr>
<td>compare_rows ........................................ 9</td>
</tr>
<tr>
<td>convert2hex ......................................... 10</td>
</tr>
<tr>
<td>crayons ............................................... 11</td>
</tr>
<tr>
<td>done .................................................. 11</td>
</tr>
<tr>
<td>dotplot ............................................. 12</td>
</tr>
<tr>
<td>errors2pushbullet ......................... 13</td>
</tr>
<tr>
<td>excel_fig ........................................... 13</td>
</tr>
<tr>
<td>exit ............................................... 14</td>
</tr>
<tr>
<td>fac2num ............................................. 15</td>
</tr>
<tr>
<td>fisher ............................................... 15</td>
</tr>
<tr>
<td>grayplot ............................................ 16</td>
</tr>
<tr>
<td>h ..................................................... 17</td>
</tr>
<tr>
<td>hex2dec ............................................ 18</td>
</tr>
<tr>
<td>histlines ........................................... 19</td>
</tr>
<tr>
<td>jiggle ............................................... 20</td>
</tr>
<tr>
<td>kbdate .............................................. 21</td>
</tr>
<tr>
<td>lenuniq ............................................. 22</td>
</tr>
<tr>
<td>make ................................................ 22</td>
</tr>
<tr>
<td>manyboxplot ....................................... 23</td>
</tr>
<tr>
<td>maxabs .............................................. 24</td>
</tr>
<tr>
<td>mypairs ............................................. 25</td>
</tr>
<tr>
<td>myround ............................................. 25</td>
</tr>
<tr>
<td>normalize .......................................... 26</td>
</tr>
<tr>
<td>note ................................................ 27</td>
</tr>
<tr>
<td>numbers ............................................. 28</td>
</tr>
<tr>
<td>objectsizes ........................................ 28</td>
</tr>
<tr>
<td>openfile ........................................... 29</td>
</tr>
<tr>
<td>paired.perm.test ................................. 30</td>
</tr>
<tr>
<td>paste .............................................. 31</td>
</tr>
<tr>
<td>paste00 ............................................. 31</td>
</tr>
<tr>
<td>perm.test .......................................... 32</td>
</tr>
<tr>
<td>plot_crayons ....................................... 33</td>
</tr>
<tr>
<td>pushbullet_devices ......................... 34</td>
</tr>
<tr>
<td>qqline2 ............................................. 34</td>
</tr>
<tr>
<td>qr2 .................................................. 35</td>
</tr>
<tr>
<td>quantileSE .......................................... 36</td>
</tr>
<tr>
<td>revrainbow .......................................... 37</td>
</tr>
<tr>
<td>rmvn ............................................... 37</td>
</tr>
<tr>
<td>runningmean ........................................ 38</td>
</tr>
<tr>
<td>runningratio ........................................ 39</td>
</tr>
<tr>
<td>setRNGparallel .................................... 40</td>
</tr>
<tr>
<td>simp ............................................... 41</td>
</tr>
<tr>
<td>stop_sending_errors ....................... 42</td>
</tr>
<tr>
<td>strwidth2lines .................................... 42</td>
</tr>
<tr>
<td>strwidth2xlim .................................... 43</td>
</tr>
<tr>
<td>switchv ............................................ 44</td>
</tr>
<tr>
<td>theme_karl ......................................... 45</td>
</tr>
</tbody>
</table>
add_commas

Description

Convert a number to a string, with commas every 3rd digit

Usage

add_commas(numbers)

Arguments

numbers Vector of non-negative numbers (will be rounded to integers)

Value

Character string with numbers written like "7,547,085".

Examples

add_commas(c(231, 91310, 2123, 991101020, 999723285))

arrowlocator

Use the locator function to plot an arrow

Description

Use the locator function to indicate the endpoints of an arrow and then plot it.

Usage

arrowlocator(reverse = FALSE, horizontal = FALSE, vertical = FALSE, length = 0.1, ...)
**Arguments**

- **reverse**: If FALSE, first indicate the tail of the arrow and then the head; if TRUE, first indicate the head of the arrow and then the tail.
- **horizontal**: If TRUE, force the arrow to be horizontal. (Use the average y-axis value of the two clicks for the vertical placement.)
- **vertical**: If TRUE, force the arrow to be vertical. (Use the average x-axis value of the two clicks for the horizontal placement.)
- **length**: Length of the edges of the arrow head.
- **...**: Additional graphics parameters

**Details**

Use `locator` to indicate the two endpoints of an arrow and then draw it.

**Value**

The locations of the endpoints of the arrow, as a two-row matrix. The first row indicates the location of the tail of the arrow; the second row indicates the location of the head of the arrow.

**See Also**

`arrows`, `locator`

**Examples**

```r
## Not run:
plot(0,0,type="n", xlab="", ylab="", xlim=c(0,100), ylim=c(0,100))
arrowslocator(col="blue", lwd=2)

## End(Not run)
```

---

### attrnames

*Get names of attributes*

**Description**

Get the names of the attributes of an object

**Usage**

`attrnames(object)`

**Arguments**

- **object**: Any object
brocolors

Details

It just does names(attributes(object)).

Value

Vector of character strings with the names of the attributes.

Examples

x <- matrix(1:100, ncol=5)
colnames(x) <- LETTERS[1:5]
attnames(x)

brocolors  Vectors of colors for figures

Description

Creates different vectors of related colors that may be useful for figures.

Usage

brocolors(set = c("general", "general2", "bg", "bgpng", "CC", "CCalt", "f2", 
"sex", "main", "crayons", "web"))

Arguments

set

Character string indicating a set of colors.

Value

Vector of character strings representing the chosen set of colors, in RGB.

See Also

plot_crayons

Examples

par(mar=c(0.6,5.1,0.6,0.6))
plot(0, 0, type="n", xlab="", ylab="", xlim=c(0, 9), ylim=c(8.5, 0), yaxs="i", 
xaxt="n", yaxt="n", xaxs="i")
axis(side=2, at=1:8, c("general", "general2", "bg", "bgpng", "CC", "f2", "sex", "main"), las=1)

gen <- brocolors("general")
points(seq(along=gen), rep(1,length(gen)), pch=21, bg=gen, cex=4)
text(seq(along=gen), rep(c(0.55, 0.7), length(gen))[seq(along=gen)], names(gen))

gen2 <- brocolors("general2")
bromanversion

Description

Print the version number of the currently installed version of R/broman.

Usage

bromanversion()

Value

A character string with the version number of the currently installed version of R/broman.

Examples

bromanversion()
**Description**

Check whether two objects are the same, including their patterns of NAs.

**Usage**

cf(a, b)

**Arguments**

a Some object.

b Another object

**Details**

It’s not very complicated: ((is.na(a) & is.na(b)) | (!is.na(a) & !is.na(b) & a == b))

**Value**

Boolean object with TRUE indicating an element is the same.

**Examples**

```r
x <- c(5, 8, 9, NA, 3, NA)
y <- c(5, 2, 9, 4, NA, NA)
cf(x,y)

x <- matrix(rnorm(1000), ncol=20)
x[sample(seq(along=x), 100)] <- NA
all(cf(x,x))
dim(cf(x,x))

y <- x
y[4,8] <- NA
sum(cf(x,y))
y[6,2] <- 18
sum(cf(x,y))
y[6,5] <- 32
sum(cf(x,y))

x <- as.data.frame(x)
y <- as.data.frame(y)
sum(cf(x,y))

x <- as.list(x)
y <- as.list(y)
```
chisq  

Chi-square test by simulation for a two-way table

Description

Calculate a p-value for a chi-square test by Monte Carlo simulation.

Usage

chisq(tab, n.sim = 1000)

Arguments

tab  A matrix of counts.
n.sim  Number of samples of permuted tables to consider.

Details

This is like the function chisq.test, but calculates an approximate P-value rather than referring to asymptotics. This will be better for large, sparse tables.

Value

A single number: the P-value testing independence of rows and columns in the table.

See Also

chisq.test, fisher.test, fisher

Examples

TeaTasting <- matrix(c(3,1,1,3),nrow=2)
chisq(TeaTasting,1000)
colwalpha

Convert a color to use alpha transparency

Description
Convert a color to RGB and then to RGB with alpha transparency

Usage
```
colwalpha(color, alpha = 1)
```

Arguments
- **color**: A character string for a color
- **alpha**: Transparency value (between 0 and 1)

Value
A character string representing a color

Examples
```
colwalpha(c("blue", "red"), 0.5)
```

compare_rows

Compare rows in a matrix

Description
For all pairs of rows in a matrix, calculate the proportion of mismatches or the RMS difference.

Usage
```
compare_rows(mat, method = c("prop_mismatches", "rms_difference"))
```

Arguments
- **mat**: Numeric matrix. Should be integers in the case method="prop_mismatches".
- **method**: Indicates whether to use proportion mismatches or the RMS difference. Missing values are omitted.

Value
A square matrix of dimension `nrow(mat)` with NAs on the diagonal and the calculated statistic in the body.
Examples

```r
n <- 10
p <- 200
x <- matrix(sample(1:4, n*p, replace=TRUE), ncol=p)
d <- compare_rows(x)
```

---

**convert2hex**

*Convert decimal to hex*

**Description**

Convert a number to hexadecimal notation.

**Usage**

```r
convert2hex(d)
```

**Arguments**

- `d` A number.

**Details**

Nothing important to say here.

**Value**

A character string; the input in hex.

**See Also**

*hex2dec*

**Examples**

```r
convert2hex(333)
dec2hex(333)
dec2hex(333) == "14D"
dec2hex(0:30)
```
crayons

Crayon colors

Description

Vector of colors corresponding to Crayola crayons

Usage

crayons(color_names = NULL, ...)

Arguments

color_names  Optional vector of color names; can be partial matches.
...  Additional optional color names

Value

Vector of named RGB colors

References


See Also

plot_crayons, brocolors

done  Send a short message via RPushbullet.

Description

Send a short message via RPushbullet, to be used to indicate that some R job is complete. #'

Usage

done(message = "R is done", recipients = NULL)

Arguments

message  A character string with a message. (passed to pbPost.)
recipients  A character or numeric vector indicating the devices this post should go to. If NULL, the default device is looked up from an optional setting, and if none has been set the push is sent to all devices. (passed to pbPost.)
Examples

### Not run: done("Your R job is complete.")

---

**dotplot**

*Dot chart with a gray background*

---

**Description**

Like the *grayplot* function, but with one axis assumed to be categorical.

**Usage**

```
dotplot(group, y, jiggle = NULL, rotate = FALSE, ...)
```

**Arguments**

- `group` Categorical coordinates for the plot
- `y` Coordinates of points in the plot
- `jiggle` Vector of amounts to jiggle the points horizontally, or a character string ("fixed" or "random") indicating the jiggling method; see `jiggle`.
- `rotate` If TRUE, have group as y-axis; default (FALSE) has group on x-axis.
- `...` Optional graphics arguments

**Details**

Calls *grayplot* with special choices of graphics parameters for the case of categorical x.

**Value**

None.

**See Also**

*grayplot*,

**Examples**

```r
x <- rnorm(40, c(1,3))
g <- rep(c("A", "B"), 20)
dotplot(g, x)
dotplot(g, x, "random")
dotplot(g, x, runif(length(g), -0.25, 0.25))
```
errors2pushbullet

Send further errors to pushbullet

Description

Set options to use RPushbullet to use pushbullet to push notifications of any error messages.

Usage

errors2pushbullet(recipients = NULL)

Arguments

recipients A character or numeric vector indicating the devices this post should go to. If NULL, the default device is looked up from an optional setting, and if none has been set the push is sent to all devices. (passed to pbPost.)

See Also

stop_sending_errors

Examples

## Not run: errors2pushbullet()

excel_fig

Excel-style figure displaying contents of a matrix

Description

Turn a matrix of data into an SVG of how it might look in Excel

Usage

excel_fig(mat, file = NULL, cellwidth = 80, cellheight = 26, textsize = 16, fig_width = NULL, fig_height = NULL, border = "#CECECE", headcol = "#E9E9E9", headborder = "#969696", headtextcol = "#626262", textcol = "black", row_names = FALSE, col_names = TRUE, hilitcells = NULL, hilitcolor = "#F0DDB", lwd = 1, direct2svg = FALSE, mar = rep(0.1, 4))
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>A matrix</td>
</tr>
<tr>
<td>file</td>
<td>Optional file name (must have extension .svg, .png, .jpg, or .pdf)</td>
</tr>
<tr>
<td>cellwidth</td>
<td>Width of each cell, in pixels</td>
</tr>
<tr>
<td>cellheight</td>
<td>Height of each cell, in pixels</td>
</tr>
<tr>
<td>textsize</td>
<td>Size for text (if file is provided or direct2svg=TRUE)</td>
</tr>
<tr>
<td>fig_width</td>
<td>Width of figure, in pixels (if NULL, taken from cellwidth); ignored when</td>
</tr>
<tr>
<td></td>
<td>direct2svg=FALSE</td>
</tr>
<tr>
<td>fig_height</td>
<td>Height of figure, in pixels (if NULL, taken from cellheight); ignored when</td>
</tr>
<tr>
<td></td>
<td>direct2svg=FALSE</td>
</tr>
<tr>
<td>border</td>
<td>Color of border of cells for the body of the matrix</td>
</tr>
<tr>
<td>headcol</td>
<td>Background color of cells on the top and left border</td>
</tr>
<tr>
<td>headborder</td>
<td>Color of border of cells on the top and left border</td>
</tr>
<tr>
<td>headtextcol</td>
<td>Color of text in cells on the top and left border</td>
</tr>
<tr>
<td>textcol</td>
<td>Color of text in cells in body of the matrix</td>
</tr>
<tr>
<td>row_names</td>
<td>If TRUE, and row names are present, include them as a first column</td>
</tr>
<tr>
<td>col_names</td>
<td>If TRUE, and column names are present, include them as a first row</td>
</tr>
<tr>
<td>hilitcells</td>
<td>Optional character vector of cells to highlight, like &quot;A1&quot; or &quot;D4&quot;</td>
</tr>
<tr>
<td>hilitcolor</td>
<td>Color to highlight cells, a vector of length 1 or the same length as hilitcells</td>
</tr>
<tr>
<td>lwd</td>
<td>Line width for rectangles</td>
</tr>
<tr>
<td>direct2svg</td>
<td>If TRUE, rather than R graphics, just print an SVG directly with cat.</td>
</tr>
<tr>
<td>mar</td>
<td>Plot margins, passed to par</td>
</tr>
</tbody>
</table>

Examples

df <- data.frame(id = c(101, 102, 103),  
                 sex = c("M", "F", "M"),  
                 weight=c(22.3, 15.8, 19.7),  
                 stringsAsFactors=FALSE)  
excel_fig(df, col_names=TRUE)

exit

exit R without saving

Description

exit R without saving workspace.

Usage

exit()
**fac2num**

*Convert a factor to numeric*

**Details**
This just calls `q("no")`

**Value**
None.

---

**fisher**

*Fisher's exact test for a two-way table*

**Description**
Performs a sampling version of Fisher's exact test for a two-way contingency table.

**Usage**

```r
fisher(tab, n.sim = 1000)
```

**Arguments**
- `tab` : A matrix of counts.
- `n.sim` : Number of samples of permuted tables to consider.
Details

This is like the function `fisher.test`, but calculates an approximate P-value rather than performing a complete enumeration. This will be better for large, sparse tables.

Value

A single number: the P-value testing independence of rows and columns in the table.

See Also

`chisq.test`, `fisher.test`, `chisq`

Examples

```r
TeaTasting <- matrix(c(3,1,1,3),nrow=2)
fisher(TeaTasting,1000)
```

---

**grayplot**

*Plot with a gray background*

Description

Like the plot function, but using a gray background just for the plot region.

Usage

```r
grayplot(x, y = NULL, ..., type = "p", hlines = NULL,
    hlines.col = "white", hlines.lty = 1, hlines.lwd = 1, vlines = NULL,
    vlines.col = "white", vlines.lty = 1, vlines.lwd = 1, xat = NULL,
    yat = NULL, bgcolor = "gray80", v_over_h = FALSE)
```

Arguments

- `x`: Coordinates of points in the plot
- `y`: Coordinates of points in the plot (optional)
- `...`: Optional graphics arguments
- `type`: Plot type (points, lines, etc.)
- `hlines`: Locations of horizontal grid lines; use `hlines=NA` to prevent horizontal grid lines
- `hlines.col`: Colors of horizontal grid lines
- `hlines.lty`: Line type of horizontal grid lines
- `hlines.lwd`: Line width of horizontal grid lines
- `vlines`: Locations of vertical grid lines; use `vlines=NA` to prevent vertical grid lines
- `vlines.col`: Colors of vertical grid lines
vlines.lty     Line type of vertical grid lines
vlines.lwd     Line width of vertical grid lines
xat            Locations for x-axis labels; xat=NA indicates no labels
yat            Locations for y-axis labels; yat=NA indicates no labels
bgcolor        Background color
v_over_h       If TRUE, place vertical grid lines on top of the horizontal ones.

Details

Calls plot with type="n", then rect to get the background, and then points. Additional arguments you can include: mgp.x and mgp.y (like mgp, for controlling parameters of axis labels, but separate for x- and y-axis).

Value

None.

See Also

plot, par, rect, points

Examples

x <- rnorm(100)
y <- x+rnorm(100, 0, 0.7)
grayplot(x, y, col="blue", pch=16)
at <- seq(-3, 3)
grayplot(x, y, col="blue", pch=16, hlines=at, vlines=at)
grayplot(x, col="violet", pch=16, bgcolor="gray90",
         hlines=seq(-4, 4, by=0.5), hlines.lwd=c(3,1),
         vlines=seq(0, 100, by=5), vlines.lwd=c(3,1,1,1))

---

h View html version of help file

Description

View the html version of a help file while running R via ESS within emacs.

Usage

h(...)

Arguments

... Help topics.
hex2dec

Details
This just calls the function help using the argument htmlhelp=TRUE.

Value
No return value.

See Also
help, help.start

Examples
h(read.cross)

Description
Convert a number from hexadecimal to decimal notation.

Usage
hex2dec(h)

Arguments
h # Character string with hexadecimal representation of a number

Details
Nothing important to say here.

Value
The input converted from hexadecimal to decimal notation.

Author(s)
Karl W Broman, <kbroman@biostat.wisc.edu>

See Also
dec2hex
Examples

hex2dec("14D")
hex2dec("14D") == 333
hex2dec(0:30)

histlines

Utility to create line-based histogram

Description

Utility function to plot histogram with lines.

Usage

histlines(x, y = NULL, breaks, use = c("counts", "density"))

Arguments

x
Either vector of breaks or the data itself.
y
Optional vector of density/counts, with length = length(x)-1.
breaks
Breaks for histogram, if y is not provided.
use
Whether to use counts or density, if y is not provided.

Details

If x and y are both provided, x is interpreted to be the breaks for a histogram, and y is a vector of counts or density values for each interval. These are then revised so that they may be plotted with lines. If y is NULL, x is taken to be the data. In this case hist is called with breaks=breaks, and either the counts or density are used as y.

Value

A data.frame with two columns: x and y.

See Also

hist, lines
jiggle

Examples

x <- rnorm(1000, mean=20, sd=5)
# basic use
out <- hist(x, breaks=60, plot=FALSE)
plot(histlines(out$breaks, out$counts),
     type="l", lwd=2, xlab="x", ylab="counts", las=1)
# alternative use
plot(histlines(x, breaks=60, use="density"),
     type="l", lwd=2, xlab="x", ylab="Density", las=1)

# comparing two distributions
z <- rnorm(1000, mean=25, sd=5)
br <- seq(min(c(x,z)), max(c(x,z)), len=50)
xlines <- histlines(x, breaks=br, use="density")
zlines <- histlines(z, breaks=br, use="density")
ymx <- max(c(xlines$y, zlines$y))*1.05
plot(xlines, ylim=c(0, ymx), yaxs="i", xaxs="i",
     type="l", lwd=2, xlab="x", ylab="Density", las=1,
     col="blue")
lines(zlines, lwd=2, col="red")

jiggle

Description

Spread points out horizontally so that, in dot plot of quantitative response in multiple categories, the separate points can be seen.

Usage

jiggle(group, y, method = c("fixed", "random"), hnum = 35, vnum = 40)

Arguments

group Categorical variable defining group; can be a factor, character, or numeric vector
y Vector of quantitative responses
method What method to use for horizontal jiggling.
hnum Number of horizontal bins for the jiggling.
vnum Number of vertical bins for the jiggling.

Details

The "random" method is similar to jitter but with amount of jiggling proportional to the number of nearby points. The "fixed" method is similar to the beeswarm package
Value

Numeric vector with amounts to jiggle the points horizontally

See Also

jitter, dotplot

---

`kbdate`  
*My little date facility*

Description

Sys.Date as a string, in a few different formats

Usage

`kbdate(format = c("dateonly", "standard"), date = Sys.time())`

Arguments

- `format` : The format for the output
- `date` : The date/time to convert

Value

A character string representation of the date/time

See Also

`Sys.time, date`

Examples

- `kbdate()
- `kbdate("standard")`
### lenuniq

**Number of unique values**

**Description**

Get the number of unique values in a vector

**Usage**

lenuniq(vec, na.rm = TRUE)

**Arguments**

- `vec`: A vector
- `na.rm`: If TRUE, remove any missing values

**Details**

It just does `length(unique(vec))` or, if `na.rm=TRUE` (the default) `length(unique(vec[!is.na(vec)]))`

**Value**

Number of unique values.

**Examples**

```r
x <- c(1, 2, 1, 3, 1, 2, 2, 3, NA, NA, 1)
lenuniq(x)
lenuniq(x, na.rm=FALSE)
```

### make

**Run make within a package directory**

**Description**

Run make within a package directory

**Usage**

make(pkg = ".", makefile = "Makefile", target = "," , quiet = FALSE)
manyboxplot

Arguments

pkg
Path to directory containing the GNU Make file, or an R package description, which can be a path or a package name. (See as.package for more information.)

makefile
File name of makefile.

target
Optional character string specifying the target.

quiet
If TRUE suppresses output from this function.

Value

Exit value from system with intern=FALSE

See Also

load_all

Examples

```r
## Not run: make() # run make within working directory
make("/path/to/mypackage") # run make within /path/to/mypackage

## End(Not run)
```

manyboxplot

Boxplot-like figure for many groups

Description

Boxplot-like figure for many groups, with lines connecting selected quantiles.

Usage

```r
manyboxplot(x, probs = c(0.05, 0.1, 0.25), dotcol = "blue",
             linecol = c("black", "red", "green", "orange"), ...)
```

Arguments

x
Matrix of data, with columns indicating the groups.

probs
Numeric vector of probabilities with values in [0,1). Quantiles will be symmetric, and the median will always be included.

dotcol
Color for median

linecol
Line colors, same length as probs

... Additional graphics parameters
Details

Calculates quantiles of the columns of \( x \) and then plots dots or lines at median plus lines at a series of quantiles, using \texttt{grayplot} for the actual plot.

Value

None.

See Also

\texttt{grayplot}

Examples

\begin{verbatim}
mu <- c(rnorm(50, 0, 0.3), rnorm(50, 2, 0.3)) # vector of means
x <- t(matrix(rnorm(1000*100, mu), ncol=1000))
manyboxplot(x, c(0.05, 0.25), ylim=range(x),
            dotcol=c("blue","green")[(1:100 > 50) + 1],
            hlines=seq(-4, 6, by=2),
            vlines=c(1, seq(20, 100, by=20)))
\end{verbatim}

Description

Take the maximum of the absolute values of the input

Usage

\texttt{maxabs(x, na.rm = FALSE)}

Arguments

- \( x \): a numeric vector or array
- \( \text{na.rm} \): a logical indicating whether missing values should be removed.

Value

The maximum of the absolute value of the input

Examples

\begin{verbatim}
x <- c(5, -2, 8, -20, 2.3)
maxabs(x)
\end{verbatim}
mypairs

My scatterplot matrix

Description

A matrix of scatterplots is produced; it’s similar to pairs, but with only the upper triangle is made.

Usage

mypairs(x, ...)

Arguments

x A numeric matrix or data frame.

... Passed to the plot function.

Details

This is like the function pairs, but only the upper triangle is produced.

Value

None.

See Also

pairs

Examples

v <- rbind(c(1,0.5,0.2),c(0.5,1,0.9),c(0.2,0.9,1))
x <- rmvn(500, rep(5,3), v)
mypairs(x, col=sample(c("blue","red"), 500, repl=TRUE))

myround

Round a number, preserving extra 0’s

Description

Round a number, preserving extra 0’s.

Usage

myround(x, digits = 1)
Arguments

- **x**: Number to round.
- **digits**: Number of digits past the decimal point to keep.

Details

Uses `sprintf` to round a number, keeping extra 0’s.

Value

A vector of character strings.

See Also

`round`, `sprintf`

Examples

```r
myround(51.01, 3)
myround(0.199, 2)
```

---

**normalize**  

**Quantile normalization**

Description

Quantile normalizes two vectors or a matrix.

Usage

```r
normalize(x, y = NULL)
```

Arguments

- **x**: Numeric vector or matrix
- **y**: Optional second numeric vector

Details

We sort the columns, take averages across rows, and then plug the averages back into the respective positions. The marginal distributions in the columns are thus forced to be the same. Missing values, which can result in differing numbers of observed values per column, are dealt with by linear interpolation.
Value

If two vectors, \( x \) and \( y \), are provided, the output is a matrix with two columns, with the quantile normalized versions of \( x \) and \( y \). If \( y \) is missing, \( x \) should be a matrix, in which case the output is a matrix of the same dimensions with the columns quantile normalized with respect to each other.

Examples

```r
z <- rmvn(10000, mu=c(0.5,10), V = rbind(c(1,0.5,0.5), c(0.5,1,0.5), c(0.5,0.5,1)))
z[sample(prod(dim(z)), 1500)] <- NA
pairs(z)
br <- seq(min(z, na.rm=TRUE), max(z, na.rm=TRUE), length=200)
par(mfrow=c(3,1))
for(i in 1:3)
  hist(z[,i], xlab="z", main=i, breaks=br)
zn <- normalize(z)
br <- seq(min(zn, na.rm=TRUE), max(zn, na.rm=TRUE), length=200)
for(i in 1:3)
  hist(zn[,i], xlab="normalized z", main=i, breaks=br)
pairs(zn)
```

---

### note

**Send a note to pushbullet**

**Description**

Even simpler interface for `pbpost` to send a bit of text to pushbullet.

**Usage**

```r
note(title, recipients = NULL, body = NULL)
```

**Arguments**

- `title` The title of the note (could be the whole thing).
- `recipients` A character or numeric vector indicating the devices this post should go to. If `NULL`, the default device is looked up from an optional setting, and if none has been set the push is sent to all devices. (passed to `pbPost`.)
- `body` The body of the note (by default, empty)

**Examples**

```r
## Not run: note("Hello.")
```
numbers | *Numbers spelled out in English*
---|---

**Description**

The numbers 1-20 spelled out in English, for use in reports.

**Format**

A vector of character strings

**Details**

- numbers - lower case
- Numbers - Capitalized

**Examples**

```r
numbers[5]
Numbers[5]
```

**objectsizes | *Calculate sizes of all objects in workspace***

**Description**

Calculate the sizes of all of the objects in one’s workspace.

**Usage**

```r
objectsizes(obj = NULL, sortbysize = TRUE)
```

**Arguments**

- **obj** Vector of object names. If missing, we pull out all object names.
- **sortbysize** If TRUE, sort the objects from smallest to largest.

**Details**

Calls `object.size` repeated to get the size of a list of objects.

**Value**

A data frame with the only column being the size of each object in megabytes (Mb). The row names are the names of the objects.
openfile

See Also

object.size, objects

Examples

print(output <- objectsizes())
## Not run: sum(output)

---

openfile  

Open a file

Description

Open a file using system and open.

Usage

openfile(file)

Arguments

file  

File name (character string)

Details

I’d thought that to open a file you’d use open in MacOS and start in Windows, but system("start myfile.pdf") doesn’t work in Windows, but rather system("open myfile.pdf") does, so here we’re just using

Value

None.

Examples

## Not run: openfile("myplot.pdf")
paired.perm.test  

Paired permutation t-test

Description

Calculates a p-value for a paired t-test via permutations.

Usage

paired.perm.test(d, n.perm = NULL, pval = TRUE)

Arguments

d  A numeric vector (of differences).

n.perm  Number of permutations to perform. If NULL, all possible permutations are considered, and an exact p-value is calculated.

pval  If TRUE, return just the p-value. If FALSE, return the actual permutation results (with the observed statistic as an attribute, "tobs").

Details

This calls the function t.test to calculate a t-statistic comparing the mean of d to 0. Permutations are performed to give an exact or approximate conditional p-value.

Value

If pval=TRUE, the output is a single number: the P-value testing for the symmetry about 0 of the distribution of the population from which d was drawn. If pval=FALSE, the output is a vector of the t statistics from the permutations. An attributed "tobs" contains the t statistic with the observed data.

See Also

t.test, perm.test

Examples

x <- c(43.3, 57.1, 35.0, 50.0, 38.2, 31.2)
y <- c(51.9, 95.1, 90.0, 49.7, 101.5, 74.1)
paired.perm.test(x-y)
**paste.**  

---

**paste with dot separator**  

---

**Description**  

Calls `paste` with `sep="."`.

**Usage**  

```r
paste(...)  
```

**Arguments**  

```r
...  
```

Passed to `paste`.

**Details**  

There’s not much to this function. It just is `paste` with `sep="."`, `'cause I’m lazy.

**Value**  

A character string or vector of character strings.

**See Also**

`paste, paste0, paste00, paste..., paste0., paste0`  

**Examples**

```r
x <- 3  
y <- 4  
paste.(x, y)  
```

---

**paste00**  

---

**paste with null or dot as separator and with collapse**

---

**Description**  

Call `paste` with `sep="." or sep="" and collapse="" or collapse="."`.

**Usage**  

```r
paste00(...)  
```
perm.test

Arguments
... Passed to paste.

Details
There's not much to these functions. paste00(...) is like paste(..., sep="", collapse="")
paste.0(...) is like paste(..., sep=".", collapse=".") paste0. (..) is like paste(..., sep="", collapse=".")
paste.0(...) is like paste(..., sep=".", collapse="")

Value
A character string or vector of character strings.

See Also
paste0, paste0, paste.

Examples
x <- c(3, 4)
y <- c(5, 6)
paste00(x, y)
paste.(x, y)
paste0.(x, y)
paste.0(x, y)

---

perm.test: Permutation t-test

Description
Calculates a p-value for a t-test via permutations.

Usage
perm.test(x, y, n.perm = NULL, var.equal = TRUE, pval = TRUE)

Arguments

x A numeric vector.
y A second numeric vector.
n.perm Number of permutations to perform. If NULL, all possible permutations are considered, and an exact p-value is calculated.
var.equal A logical variable indicating whether to treat the two population variances as being equal.
pval If TRUE, return just the p-value. If FALSE, return the actual permutation results (with the observed statistic as an attribute, "tobs").
Details

This calls the function `t.test` to calculate a t-statistic comparing the vectors \( x \) and \( y \). Permutations are performed to give an exact or approximate conditional p-value.

Value

If `pval=TRUE`, the output is a single number: the P-value testing for a difference in the distributions of the populations from which \( x \) and \( y \) were drawn. If `pval=FALSE`, the output is a vector of the t statistics from the permutations. An attributed "tobs" contains the t statistic with the observed data.

See Also

`t.test`, `paired.perm.test`

Examples

```
x <- c(43.3, 57.1, 35.0, 50.0, 38.2, 61.2)
y <- c(51.9, 95.1, 90.0, 49.7, 101.5, 74.1)
perm.test(x, y)
```

---

**plot_crayons**  
*Illustration of crayon colors*

Description

Creates a plot of the crayon colors in `brocolors`

Usage

```
plot_crayons(method2order = c("hsv", "cluster"), cex = 0.6, mar = rep(0.1, 4), bg = "white", fg = "black", border = FALSE)
```

Arguments

- `method2order`: method to order colors ("hsv" or "cluster")
- `cex`: character expansion for the text
- `mar`: margin parameters: vector of length 4 (see `par`)
- `bg`: Background color
- `fg`: Foreground color (for text and box outlines)
- `border`: If TRUE, plot a border around each rectangle

Value

None
References


See Also

brocolors

Examples

plot_crayons()

pushbullet_devices  
Grab info on Pushbullet devices.

Description

Get names and identifiers of Pushbullet devices.

Usage

pushbullet_devices()

Value

data frame with nickname, model, and iden for active devices.

Description

Adds a line to a quantile-quantile plot for two datasets, from stats[qqplot]. (The available qqline function works mainly for qqnorm, with one sample being theoretical quantiles.)

Usage

qqline2(x, y, probs = c(0.25, 0.75), qtype = 7, ...)

Arguments

x  The first sample
y  The second sample.
probs  numeric vector of length two, representing probabilities. Corresponding quantile pairs define the line drawn.
qtype  the type of quantile computation used in quantile.
...  graphical parameters.
Value

Intercept and slope of the line.

See Also

qqline, qqplot

Examples

```r
x <- rchisq(500, 3)
y <- rgamma(730, 3, 1/2)
qqplot(x, y)
qqline2(x, y)
```

---

**qr2**  
*The QR decomposition of a matrix*

Description

Computes the QR decomposition of a matrix.

Usage

```r
qr2(x, tol = 0.0000001)
```

Arguments

- `x`: A matrix whose QR decomposition is to be computed.
- `tol`: The tolerance for detecting linear dependencies in the columns of `x`.

Details

Calls the function `qr` and returns less compact but more understandable output.

Value

A list of two matrices: Q and R.

See Also

`qr`

Examples

```r
hilbert <- function(n) { i <- 1:n; 1/outer(i-1,i,"+") }
h5 <- hilbert(5);
qr2(h5)
```
quantileSE

Sample quantiles and their standard errors

Description

Calculate sample quantiles and their estimated standard errors.

Usage

quantileSE(x, p = 0.95, bw = NULL, na.rm = TRUE, names = TRUE)

Arguments

- **x**: Numeric vector whose sample quantiles are wanted.
- **p**: Numeric vector with values in [0,1].
- **bw**: Bandwidth to use in the density estimation.
- **na.rm**: Logical; if true, and NA and NaN's are removed from x before the quantiles are computed.
- **names**: Logical; if true, the column names of the result is set to the values in p.

Details

The sample quantiles are calculated with the function quantile. Standard errors are obtained by the asymptotic approximation described in Cox and Hinkley (1974). Density values are estimated using a kernel density estimate with the function density.

Value

A matrix of size 2 x length(p). The first row contains the estimated quantiles; the second row contains the corresponding estimated standard errors.

See Also

quantile, density

Examples

quantileSE(rchisq(1000,4), c(0.9,0.95))
revrainbow

Create vector of colors from blue to red

Description
Calls `rainbow` then `rev`

Usage
```
revrainbow(n = 256, ...)
```

Arguments
- `n` Number of colors.
- `...` Passed to `rainbow`.

Details
There’s not much to this. It’s just `rev(rainbow(start=0, end=2/3, ...))`.

Value
Vector of colors, from blue to red.

See Also
- `rev`, `rainbow`

Examples
```
x <- matrix(rnorm(100), ncol=10)
image(x, col=revrainbow())
```

rmvn

Simulate multivariate normal

Description
Simulate from a multivariate normal distribution.

Usage
```
rmvn(n, mu = 0, V = matrix(1))
```
Arguments

n  Number of simulation replicates.
mu  Mean vector.
v  Variance-covariance matrix.

Details

Uses the Cholesky decomposition of the matrix V, obtained by chol.

Value

A matrix of size n x length(mu). Each row corresponds to a separate replicate.

See Also

rnorm

Examples

x <- rmvn(100, c(1,2), matrix(c(1,1,1),ncol=2))

runningmean  Running mean, sum, or median

Description

Calculates a running mean, sum or median with a specified window.

Usage

runningmean(pos, value, at = NULL, window = 1000, what = c("mean", "sum", "median", "sd"))

Arguments

pos  Positions for the values.
value  Values for which the running mean/sum/median/sd is to be applied.
at  Positions at which running mean (or sum or median or sd) is calculated. If NULL, pos is used.
window  Window width.
what  Statistic to use.

Value

A vector with the same length as the input at (or pos, if at is NULL), containing the running statistic.
runningratio

Author(s)
Karl W Broman <kbroman@biostat.wisc.edu>

See Also
runningratio

Examples

```r
x <- 1:10000
y <- rnorm(length(x))
plot(x,y, xaxs="i", yaxs="i")
lines(x, runningmean(x, y, window=100, what="mean"),
     col="blue", lwd=2)
lines(x, runningmean(x, y, window=100, what="median"),
     col="red", lwd=2)
lines(x, runningmean(x, y, window=100, what="sd"),
     col="green", lwd=2)
```

Description

Calculates a running ratio; a ratio sum(top)/sum(bottom) in a sliding window.

Usage

```r
runningratio(pos, numerator, denominator, at = NULL, window = 1000)
```

Arguments

- **pos**: Positions for the values.
- **numerator**: Values for numerator in ratio.
- **denominator**: Values for denominator in ratio.
- **at**: Positions at which running ratio is calculated. If NULL, pos is used.
- **window**: Window width.

Value

A vector with the same length as the input at (or pos, if at is NULL), containing the running ratio.

Author(s)
Karl W Broman <kbroman@biostat.wisc.edu>
See Also

runningmean

Examples

x <- 1:1000
y <- runif(1000, 1, 5)
z <- runif(1000, 1, 5)
plot(x, runningratio(x, y, z, window=5), type="l", lwd=2)
lines(x, runningratio(x, y, z, window=50), lwd=2, col="blue")
lines(x, runningratio(x, y, z, window=100), lwd=2, col="red")
simp

Numerical integration

Description

Perform numerical integration by Simpson’s rule or the trapezoidal rule.

Usage

simp(f, a, b, tol = 0.00000001, max.step = 1000, ...)

Arguments

f The integrand; must be a vectorized function.
a Lower limit of integration.
b Upper limit of integration.
tol Tolerance for choosing the number of grid points.
max.step Log base 2 of the total number of grid points.
... Other arguments passed to the integrand, f.

Details

Iterately doubles the number of grid points for the numerical integral, stopping when the integral decreases by less than tol.

Value

The integral of f from a to b.

See Also

integrate

Examples

f <- function(x) x**x*(1-x)*sin(x**x)
I1 <- trap(f,0,2)
I2 <- simp(f,0,2)
stop_sending_errors  
*Stop sending errors to pushbullet*

**Description**
Clear the error option, so that error notifications are no longer sent to pushbullet.

**Usage**
```r
stop_sending_errors()
```

**See Also**
- `errors2pushbullet`

**Examples**
```r
## Not run: stop_sending_errors()
```

---

`strwidth2lines`  
*Calculate width of a character string in number of lines*

**Description**
Convert stringwidth units to number of (margin) lines

**Usage**
```r
strwidth2lines(s, ...)
```

**Arguments**
- `s`  
  A character or expression vector whose length is to be calculated
- `...`  
  Additional information used by `strwidth`, such as `cex`

**Value**
Maximum string width in units of margin lines

**Author(s)**
Aimee Teo Broman
Examples

```r
p <- par(TRUE)
string <- sapply(sample(1:20,15,replace=TRUE),
    function(a) paste(LETTERS[1:a], collapse=""))
nlines <- strwidth2lines(string)
mar <- par("mar")
par(mar=c(mar[1],nlines+0.1,mar[3:4]))
plot(1:length(string),1:length(string),yaxt="n", ylab="")
    axis(side=2, at=seq_along(string), lab=string, las=1)
par(p)
nlines <- strwidth2lines(string,cex=1.5)
par(mar=c(mar[1:3],nlines+0.1))
plot(1:length(string),1:length(string),ylab="")
mgp <- par("mgp")
axis(side = 4, at=seq_along(string),
    labels = string ,las=1, hadj=1,
    mgp=c(mgp[1],nlines,mgp[3]),cex.axis=1.5)
par(p)
```

strwidth2xlim

---

**strwidth2xlim**

*Calculate horizontal limit in user coordinates for adding labels*

### Description

Calculates the x-axis limits when adding (long) labels to a plot.

### Usage

```r
strwidth2xlim(x, xstring, pos = 4, offset = 0.5, ...)```

### Arguments

- `x` numeric vector of horizontal coordinates
- `xstring` character vector, specifying text to be written
- `pos` position specifier for text; values of 1, 2, 3, and 4, respectively, indicate positions below, to the left of, above, and to the right of the coordinates
- `offset` offset of the label from the coordinate in fractions of a character width
- `...` additional text parameters from `par`, such as `cex`

### Details

See text for details on `pos` and `offset`.

### Value

Minimum and maximum x-axis limits for adding horizontal text
**switchv**

**Author(s)**
Aimee Teo Broman

**See Also**
text

**Examples**

```r
x <- runif(15, -1, 1)*10
xlabs <- sapply(sample(1:20, 15, replace=TRUE), 
    function(a) paste(LETTERS[1:a], collapse=""))
## Labels to the left ##
xlims <- strwidth2xlim(x, xlabs, pos=2)
plot(x, 1:1:length(x), xlim=xlims)
text(x, 1:1:length(x), xlabs, pos=2)
## Labels to the right ##
xlims <- strwidth2xlim(x, xlabs, pos=4, cex=0.7)
plot(x, 1:1:length(x), xlim=xlims)
text(x, 1:1:length(x), xlabs, pos=4, cex=0.7)
```

---

**Description**

Vectorized version of `switch`: just loops over input and calls `switch`.

**Usage**

```r
switchv(EXPR, ...)
```

**Arguments**

- **EXPR**
  An expression evaluating to a vector of numbers or strings
- **...**
  List of alternatives

**Value**

Vector of returned values.

**Examples**

```r
switchv(c("horse", "fish", "cat", "bug"),
    horse="fast",
    cat="cute",
    "what?")
```
theme_karl

Karl’s ggplot2 theme

Description
Karl’s ggplot2 theme: black border and no ticks

Usage
theme_karl(base_size = 12, base_family = "", ...)
karl_theme(base_size = 12, base_family = "", ...)

Arguments
base_size  Base font size
base_family Base font family
...  Passed to theme

Value
An object as returned by theme

See Also
theme

Examples
library(ggplot2)
mtcars$cyl <- factor(mtcars$cyl)
ggplot(mtcars, aes(y=mpg, x=disp, color=cyl)) +
  geom_point() + theme_karl()

triarrow
Plot an arrow within a Holmans triangle

Description
Plot an arrow within a Holmans triangle (an equilateral triangle used to depict trinomial distributions).

Usage
triarrow(x, ...)
Arguments

x  A matrix with three rows and two columns, each column being a trinomial distribution. An arrow between the two points is plotted.

...  Passed to arrows.

Details

Plot of an equilateral triangle, in order to depict trinomial distributions. A trinomial distribution (that is, a trio of non-negative numbers that add to 1) is equated to a point in the triangle through the distances to the three sides. This makes use of the fact that for any point in an equilateral triangle, the sum of the distances to the three sides is constant. First use triplot to first plot the equilateral triangle.

Value

The (x,y) coordinates of the endpoints of the arrows plotted.

See Also

triplot, tripoints, trilines

Examples

triplot()
x <- cbind(c(0.9, 0.05, 0.05), c(0.8, 0.1, 0.1), c(0.1, 0.9, 0))
tripoints(x, lwd=2, col=c("black","blue","red","green"), pch=16)
trilines(x, lwd=2, col="orange")
y <- cbind(c(0.05, 0.9, 0.05), c(0.25, 0.25, 0.5))
triarrow(y, col="blue", lwd=2, len=0.1)
**Details**

Plot of an equilateral triangle, in order to depict trinomial distributions. A trinomial distribution (that is, a trio of non-negative numbers that add to 1) is equated to a point in the triangle through the distances to the three sides. This makes use of the fact that for any point in an equilateral triangle, the sum of the distances to the three sides is constant. First use `triplot` to first plot the equilateral triangle.

**Value**

The (x,y) coordinates of the endpoints of the lines plotted.

**See Also**

`triplot`, `tripoints`, `triarrow`

**Examples**

```r
triplot()
x <- cbind(c(0.9, 0.05, 0.05), c(0.8, 0.1, 0.1), c(0.1, 0.9, 0.0), c(0, 0.9, 0.1))
tripoints(x, lwd=2, col=c("black","blue","red","green"), pch=16)
trilines(x, lwd=2, col="orange")
y <- cbind(c(0.05, 0.05, 0.9), c(0.25, 0.25, 0.5))
triarrow(y, col="blue", lwd=2, len=0.1)
```

---

**Description**

Plot Holmans triangle (an equilateral triangle used to depict trinomial distributions).

**Usage**

`triplot(labels = c("(1,0,0)", "(0,1,0)", "(0,0,1)"), ...)`

**Arguments**

- `labels` Labels for the three corners (lower-right, top, lower-left).
- `...` Passed to `plot`.

**Details**

Plot of an equilateral triangle, in order to depict trinomial distributions. A trinomial distribution (that is, a trio of non-negative numbers that add to 1) is equated to a point in the triangle through the distances to the three sides. This makes use of the fact that for any point in an equilateral triangle, the sum of the distances to the three sides is constant. The `triplot` function creates an empty triangle for use with the related functions `tripoints`, `trilines`, `triarrow`. 
Value

The (x,y) coordinates of the points plotted, if any.

See Also

triplots, trilines, triarrow

Examples

triplots()
x <- cbind(c(0.9, 0.05, 0.05), c(0.8, 0.1, 0.1), c(0.1, 0.9, 0))
triplines(x, lwd=2, col=c("black", "blue", "red", "green"), pch=16)
trilines(x, lwd=2, col="orange")
y <- cbind(c(0.05, 0.05, 0.9), c(0.25, 0.25, 0.5))
triarrow(y, col="blue", lwd=2, len=0.1)

Description

Plot points within a Holmans triangle (an equilateral triangle used to depict trinomial distributions).

Usage

tripoints(x, ...)

Arguments

x

A matrix with three rows, each column being a trinomial distribution.

... 

Passed to points.

Details

Plot of an equilateral triangle, in order to depict trinomial distributions. A trinomial distribution (that is, a trio of non-negative numbers that add to 1) is equated to a point in the triangle through the distances to the three sides. This makes use of the fact that for any point in an equilateral triangle, the sum of the distances to the three sides is constant. First use triplot to first plot the equilateral triangle.

Value

The (x,y) coordinates of the points plotted.

See Also

triplot, trilines, triarrow
Examples

```r
triplot()
x <- cbind(c(0.9, 0.05, 0.05), c(0.8, 0.1, 0.1), c(0.1, 0.9, 0.1), c(0, 0.9, 0.1))
tripoints(x, lwd=2, col=c("black","blue","red","green"), pch=16)
trilines(x, lwd=2, col="orange")
y <- cbind(c(0.05, 0.05, 0.9), c(0.25, 0.25, 0.5))
triarrow(y, col="blue", lwd=2, len=0.1)
```

Description

Plot a Venn diagram (with two groups), to scale, either with circles or with squares.

Usage

```r
venn(setA = 50, setB = 50, both = 25, method = c("circle", "square"),
labels = c("A", "B"), col = c("blue", "red"))
```

Arguments

- `setA`: Total area of set A.
- `setB`: Total area of set B.
- `both`: Area of intersection of sets A and B.
- `method`: Indicates whether to plot circles or squares.
- `labels`: Labels for the two sets. (NULL for no labels.)
- `col`: Colors of the two sets.

Details

Plots a to-scale Venn diagram with two sets, so that the relative areas of the two sets and their intersection are exact.

Value

None.

Examples

```r
venn(setA=86, setB=1622, both=10)
venn(setA=86, setB=1622, both=10, method="square")
```
## winsorize

**Winsorize a vector**

**Description**
For a numeric vector, move values below and above the q and 1-q quantiles to those quantiles.

**Usage**
```r
winsorize(x, q = 0.006)
```

**Arguments**
- **x**: Numeric vector
- **q**: Lower quantile to use

**Value**
A vector like the input `x`, but with extreme values moved in to the `q` and `1-q` quantiles.

**Examples**
```r
x <- sample(c(1:10, rep(NA, 10), 21:30))
winsorize(x, 0.2)
```

## xlimlabel

**Calculate horizontal limit in user coordinates for adding labels**

**Description**
Calculates the x-axis limits when adding (long) labels to a plot.

**Usage**
```r
xlimlabel(x, xlabels, pos = 4, offset = 0.5, ...)
```

**Arguments**
- **x**: numeric vector of horizontal coordinates
- **xlabels**: character vector, specifying text to be written
- **pos**: position specifier for text; values of 1, 2, 3, and 4, respectively, indicate positions below, to the left of, above, and to the right of the coordinates
- **offset**: offset of the label from the coordinate in fractions of a character width
- **...**: Additional par arguments
Details

See text for details on pos and offset.

Value

Minimum and maximum x-axis limits for adding horizontal text

Author(s)

Aimee Teo Broman

See Also

text

Examples

```r
x <- runif(15, -1, 1)*10
xlabs <- sapply(sample(1:20, 15, replace=TRUE),
                function(a) paste(LETTERS[1:a], collapse=""))
par(mfrow=c(2,1), las=1)
## Labels to the left ##
xlims <- xlimlabel(x, xlabs, pos=2)
plot(x, 1:length(x), xlim=xlims, ylab="Index")
text(x, 1:length(x), xlabs, pos=2)
## Labels to the right ##
xlims <- xlimlabel(x, xlabs, pos=4, cex=0.7)
plot(x, 1:length(x), xlim=xlims, ylab="Index")
text(x, 1:length(x), xlabs, pos=4, cex=0.7)
```
Index

*Topic **IO**
  openfile, 29

*Topic **algebra**
  qr2, 35

*Topic **character**
  paste., 31
  paste00, 31

*Topic **color**
  colwalpha, 9
  revrainbow, 37

*Topic **datagen**
  rmvn, 37

*Topic **datasets**
  numbers, 28

*Topic **data**
  cf, 7

*Topic **documentation**
  h, 17

*Topic **graphics**
  dotplot, 12
  grayplot, 16
  histlines, 19
  manyboxplot, 23

*Topic **hplot**
  arrowlocator, 3
  excel_fig, 13
  mypairs, 25
  qline2, 34
  triarrow, 45
  triline, 46
  triplot, 47
  tripoints, 48
  venn, 49

*Topic **htest**
  chisq, 8
  fisher, 15
  paired.perm.test, 30
  perm.test, 32

*Topic **manip**
  convert2hex, 10
  hex2dec, 18

*Topic **math**
  simp, 41

*Topic **print**
  bromanversion, 6

*Topic **univar**
  quantileSE, 36
  runningmean, 38
  runningratio, 39

*Topic **utilities**
  attrnames, 4
  brocolors, 5
  crayons, 11
  done, 11
  errors2pushbullet, 13
  exit, 14
  kbdace, 21
  lenuniq, 22
  make, 22
  myround, 25
  normalize, 26
  note, 27
  objectsizes, 28
  setRNGparallel, 40
  stop_sending_errors, 42
  winsorize, 50
  add_commas, 3
  arrowlocator, 3
  arrows, 4, 46
  as.package, 23
  attrnames, 4
  brocolors, 5, 11, 33, 34
  bromanversion, 6
  cat, 14
  cf, 7
  chisq, 8, 16
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Numbers</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>chisq.test</td>
<td>8, 16</td>
<td></td>
</tr>
<tr>
<td>chol</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>colwalpha</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>compare_rows</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>convert2hex</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>crayons</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>dec2hex</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>dec2hex (convert2hex)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>density</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>done</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>dotplot</td>
<td>12, 21</td>
<td></td>
</tr>
<tr>
<td>errors2pushbullet</td>
<td>13, 42</td>
<td></td>
</tr>
<tr>
<td>excel_fig</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>fac2num</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>fisher</td>
<td>8, 15</td>
<td></td>
</tr>
<tr>
<td>fisher.test</td>
<td>8, 16</td>
<td></td>
</tr>
<tr>
<td>grayplot</td>
<td>12, 16, 24</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>help</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>help.start</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>hex2dec</td>
<td>10, 18</td>
<td></td>
</tr>
<tr>
<td>hist</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>histlines</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>integrate</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>jiggle</td>
<td>12, 20</td>
<td></td>
</tr>
<tr>
<td>jitter</td>
<td>20, 21</td>
<td></td>
</tr>
<tr>
<td>karl_theme (theme_karl)</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>kbdate</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>lenuniq</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>lines</td>
<td>19, 46</td>
<td></td>
</tr>
<tr>
<td>load_all</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>locator</td>
<td>3, 4</td>
<td></td>
</tr>
<tr>
<td>make</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>manyboxplot</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>maxabs</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>mypairs</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>myround</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>normalize</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Numbers (numbers)</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>numbers</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>object.size</td>
<td>28, 29</td>
<td></td>
</tr>
<tr>
<td>objects</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>object.size</td>
<td>28, 29</td>
<td></td>
</tr>
<tr>
<td>openfile</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>paired.perm.test</td>
<td>30, 33</td>
<td></td>
</tr>
<tr>
<td>pairs</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>par</td>
<td>14, 17, 33</td>
<td></td>
</tr>
<tr>
<td>paste</td>
<td>31, 32</td>
<td></td>
</tr>
<tr>
<td>paste.. (paste00)</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>paste.0</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>paste.0 (paste00)</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>paste0.</td>
<td>31, 32</td>
<td></td>
</tr>
<tr>
<td>paste0..</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>paste0. (paste00)</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>paste00</td>
<td>31, 31</td>
<td></td>
</tr>
<tr>
<td>pbPost</td>
<td>11, 13, 27</td>
<td></td>
</tr>
<tr>
<td>perm.test</td>
<td>30, 32</td>
<td></td>
</tr>
<tr>
<td>plot</td>
<td>17, 25, 47</td>
<td></td>
</tr>
<tr>
<td>plot_crayons</td>
<td>5, 11, 33</td>
<td></td>
</tr>
<tr>
<td>points</td>
<td>17, 48</td>
<td></td>
</tr>
<tr>
<td>pushbullet_devices</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>qqline</td>
<td>34, 35</td>
<td></td>
</tr>
<tr>
<td>qqline2</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>qqnorm</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>qqplot</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>qr</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>qr2</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>quantile</td>
<td>34, 36</td>
<td></td>
</tr>
<tr>
<td>quantileSE</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>rainbow</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>rect</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>rev</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>revrainbow</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>rmvn</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>rnorm</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>round</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>runningmean</td>
<td>38, 40</td>
<td></td>
</tr>
<tr>
<td>runningratio</td>
<td>39, 39</td>
<td></td>
</tr>
<tr>
<td>setRNGparallel</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
simp, 41
sprintf, 26
stats, 34
stop_sending_errors, 13, 42
strwidth2lines, 42
strwidth2xlim, 43
switch, 44
switchv, 44
Sys.time, 21
system, 23, 29
t.test, 30, 33
text, 51
theme, 45
theme_karl, 45
trap(simp), 41
triarrow, 45, 47, 48
trilines, 46, 46, 47, 48
triplot, 46, 47, 47, 48
tripoints, 46–48, 48

unsetRNGparallel (setRNGparallel), 40

venn, 49

winsorize, 50

xlimlabel, 50