Package 'berryFunctions'

April 10, 2017

Type Package

Title Function Collection Related to Plotting and Hydrology

Version 1.15.0

Date 2017-04-10

Imports grDevices, graphics, stats, utils, abind

Suggests RColorBrewer, pbapply, knitr, rmarkdown, gstat, RCurl

Author Berry Boessenkool

Maintainer Berry Boessenkool <berry-b@gmx.de>

Description Draw horizontal histograms, color scattered points by 3rd dimension,
enhance date- and log-axis plots, zoom in X11 graphics, trace errors and warnings,
use the unit hydrograph in a linear storage cascade, convert lists to data.frames and arrays,
fit multiple functions.

License GPL (>= 2)

URL https://github.com/brry/berryFunctions

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-04-10 14:55:04 UTC

R topics documented:

`berryFunctions-package` *Berry's functions*

Description

Collection of functions, mainly connected with graphics and hydrology.
- zoom in X11 graphics
- plot rainfall-runoff data and optimize parameters for the unit hydrograph in the linear storage cascade
- write text to plots on top of colored fields in label size (halo-effect)
- draw scatterplots colored by 3rd dimension (as in image, which only deals with grids)
- draw histograms horizontally
- advancedly label date axes and logarithmic axes
- fit multiple functions (power, reciprocal, exponential, logarithmic, polynomial, rational) by regression
- convert lists to data.frames
- and more...

Note

Get the most recent code updates at https://github.com/brry
At some places you'll find `## not run` in the examples. These code blocks were excluded from checking while building, mainly because they are interactive and need mouseclicks, or because they open another device/file. Normally, you should be able to run them in an interactive session. If you do find unexecutable code, please tell me!
Feel free to suggest packages in which these functions would fit well.
I strongly depend on - and therefore welcome - any feedback!

The following functions have been deprecated:
changeAttribute, showAttribute, shapeZoom: moved to https://github.com/brry/shapeInteractive
extremeStat, extremeStatLmom: moved to distLextreme in https://github.com/brry/extremeStat
compFiles has been renamed to `compareFiles`. combineTextfiles has been renamed to `combineFiles`.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011-2016

Examples

```
# see   vignette("berryFunctions")
```

addAlpha *Color transparency*

Description

Make existing colors semi-transparent (add alpha)

Usage

```
addAlpha(col, alpha = 0.3)
```

Arguments

col	Vector of color names (colors), hexadecimal or integer that can be interpreted by col2rgb
alpha	Level of semi-transparency. between 0 (transparent) and 1 (intransparent). Can also be a vector. DEFAULT: 0.3

Value

character vector with hexadecimal color codes.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, June 2014 Based on suggestion by Mathias Seibert, Dec. 2013

See Also

addFade, rgb, colors, col2rgb

Examples

```
addAlpha("red", c(0.1, 0.3, 0.6, 1))
addAlpha(1:3)
addAlpha(1:3, 1:3/3)
NewColors <- addAlpha(c("red","blue","yellow","green", "purple"), 0:200/200)
plot(runif(1000), col=NewColors, pch=16, cex=2)

# use addFade for line segments, because of overlapping dots
set.seed(1); x <- cumsum(rnorm(30))  ; y <- x-2
plot(x, type="n")
segments(x0=1:29,y0=head(x,-1), x1=2:30,y1=x[-1], col=addAlpha(4, 29:0/30), lwd=10)
segments(x0=1:29,y0=head(y,-1), x1=2:30,y1=y[-1], col=addFade (4, 29:0/30), lwd=10)
```

addFade *Color fade out*

Description

Make existing colors fade away to white

Usage

```
addFade(col, fade = 0.3, target = "white", ...)
```

Arguments

col	Vector of color names (colors), hexadecimal or integer that can be interpreted by col2rgb
fade	Level of fading towards target. between 0 (target) and 1 (col). Can also be a vector. DEFAULT: 0.3
target	Target color that should be faded into. DEFAULT: "white"
...	Further arguments passed to colorRamp

Value

character matirx with hexadecimal color codes.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2016

See Also

addAlpha, colorRamp, colors

Examples

```
plot(1:11, pch=16, cex=3, col=addFade(2, 10:0/10))
plot(1:11, pch=16, cex=3, col=addFade(2, 10:0/10, target="blue"))
plot(1:11, pch=16, cex=3, col=addFade(2, 10:0/10, target=3:4))
plot(1:21, pch=16, cex=3, col=addFade(2:3, 10:0/10))
plot(1:21, pch=16, cex=3, col=addFade(2:3, 10:0/10, target=4:5))
NewColors <- addFade(c("red","blue","yellow","green", "purple"), 0:200/200)
plot(runif(1000), col=NewColors, pch=16, cex=2)
```

addRows *Add n rows to a data.frame*

Description

simple Helper-Function to add n rows to a data.frame.

Usage

```
addRows(df, n, values = NA)
```

Arguments

df	Dataframe object
n	Number of rows to add
values	Values to be used in the new rows. DEFAULT: NA

Value

A data.frame

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2014

See Also

insertRows, data.frame, matrix, rbind

Examples

```
MYDF <- data.frame(A=5:3, B=2:4)
addRows(MYDF, 3)
```

`almost.equal` *Vectorized testing for near-equality*

Description

Vectorized testing for near-equality with `all.equal`. Since elements are recycled, this will not work for environments. You *can* use `almost.equal` directly in `if` expressions.

Usage

```
almost.equal(x, y, ...)
```

Arguments

`x, y`	R objects to be compared with each other, recycled to max length
`...`	Further arguments passed to `all.equal`

Value

Logical vector

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2017

See Also

`all.equal`

Examples

```
# General usage:
x <- c(0.4-0.1, 0.5-0.2)
x
x==0.3                    # FALSE TRUE # but mathematically, x is 0.3
all.equal(x, rep(0.3,2))  # TRUE
almost.equal(x,0.3)       # TRUE TRUE  # nice

y <- c(7777, 0.3)
   all.equal(x,y) # "Mean relative difference: 25922.33"   Not what I want
almost.equal(x,y) # FALSE TRUE                             Exactly what I want


# Testing vectorization
almost.equal(1:6, 3)
almost.equal(1:6, NA)
almost.equal(1:6, NULL)
```

```
# Testing the function for different data types (in order of coercion):
almost.equal(c(TRUE,FALSE,NA), c(TRUE,FALSE,NA))      # logical
almost.equal(as.factor(letters), as.factor(letters))  # factor
   all.equal(1:6, 1:6)
almost.equal(1:6, 1:6)                                # integer  numeric see above
0.4+0.4i - 0.1-0.1i == 0.3+0.3i
almost.equal(0.4+0.4i - 0.1-0.1i, 0.3+0.3i)           # complex
   all.equal(letters, tolower(LETTERS))
almost.equal(letters, tolower(LETTERS))               # character
almost.equal(Sys.Date()+1:4,Sys.Date()+1:4)           # Date
x <- Sys.time()+0:2
all.equal(x,x)
almost.equal(x,x)                                     # POSIXt
A <- list(a=1:5, b=0.5-0.2)
B <- list(a=1:5, b=0.4-0.1)
   all.equal(A,B)
almost.equal(A,B)                                     # list
```

---

anhang *open the Appendix of Rclick*

Description

Open the Appendix of my R handbook found online at RclickHandbuch.wordpress.com

Usage

```
anhang()
```

Value

None, opens pdf in default viewer using system2

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jul 2016

See Also

funSource

Examples

```
# anhang() # excluded from cran check because of external browser opening policy
```

approx2 *Smart linear NA interpolation*

Description

Smart interpolation: as approx, approx2 fills NAs in a vector with linear interpolation, but unlike approx, it can handle NAs at the ends of a vector (takes the first/last value available for those). Also, approx2 returns a vector only.

Usage

```
approx2(x, fill = NULL, n = length(x), quiet = FALSE, ...)
```

Arguments

x	Vector with (numeric) values
fill	Function to fill NAs at the start or end of the vector. See Details. DEFAULT: NULL
n	Number of points to interpolate to
quiet	Logical: suppress warning for no non-NA values? DEFAULT: FALSE
...	Further arguments passed to approx

Details

The function fill is used to fill missing values at the ends of the vector. It could be mean or median, for example, but must be a function that accepts na.rm=TRUE as an argument. The default (NULL) means to use the first (or last) observation available.

Value

Vector with NAs replaced with interpolation (not a list, as in approx!)

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2015

See Also

approx, zoo::na.locf, ciBand for usage example

Examples

```
is.error(  approx2(c(NA,NA))  ) # yields an error
approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1))
approx2(c( 2,NA, 6, 4, 8, 9, 3, 2, 1))
approx2(c( 2, 4, 6, 4, 8, 9,NA, 2,NA))

approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1))
approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1), fill=median)
approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1), fill=mean)

approx2(c( 3, 4, 6, 4, 8, 9,NA, 2,NA))
approx2(c( 3, 4, 6, 4, 8, 9,NA, 2,NA), fill=median)
approx2(c( 3, 4, 6, 4, 8, 9,NA, 2,NA), fill=mean)

approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1), n=17)
approx2(c( 2,NA, 6, 4, 8, 9, 3, 2, 1), n=17)
approx2(c( 2, 4, 6, 4, 8, 9,NA, 2,NA), n=17)
```

---

around *View values around an index*

Description

View index rows of a data.frame with n surrounding rows

Usage

```
around(x, i, n1 = 2, n2 = n1, convert = is.logical(i))
```

Arguments

x	Data.frame
i	Index (logical or integers)
n1	Number of elements shown before each i. DEFAULT: 2
n2	Number of elements shown after each i. DEFAULT: n1
convert	Use `which` to get the row numbers? DEFAULT: TRUE if i is boolean

Value

Nothing, calls `View`

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Nov 2016

See Also

sortDF, View

Examples

```
## Not run:  ## View should not be used in examples
myDF <- data.frame(A=1:30, B=cumsum(rnorm(30)))
myDF[c(5,7,23,29),1] <- NA
around(myDF, i=is.na(myDF$A))
around(myDF, i=c(11,19), n2=0)

## End(Not run)
```

---

betaPlot *Beta density plot*

Description

Quick and nice plot of beta density distribution based on just alpha and beta

Usage

```
betaPlot(shape1 = 1.5, shape2 = 5, lines = NA, fill = rgb(0, 0.3, 0.8,
  0.4), cumulative = TRUE, mar = c(2, 3, 3, 3), keeppar = FALSE,
  las = 1, main = paste("Beta density with\nalpha =", signif(shape1, 3),
  "and beta =", signif(shape2, 3)), ylim = lim0(y), xlim = 0:1, ylab = "",
  xlab = "", type = "l", lty = 1, col = par("fg"), ...)
```

Arguments

shape1	Alpha value as in dbeta. DEFAULT: 1.5
shape2	Beta value. DEFAULT: 5
lines	Quantiles at which vertical lines should be plotted. DEFAULT: NA
fill	Color passed to polygon. DEFAULT: rgb(0,0.3,0.8, 0.4)
cumulative	Should cumulative density distribution be added? DEFAULT: TRUE
mar	Margins for plot passed to par. DEFAULT: c(2,3,3,3)
keeppar	Should margin parameters be kept instead of being restored to previous value? DEFAULT: FALSE
las	Label orientation, argument passed to plot. DEFAULT: 1
main	main as in plot. DEFAULT: paste("Beta density with\nalpha =", shape1, "and beta =", shape2)
ylim, xlim	limit for the y and x axis. DEFAULT: lim0(y), 0:1
ylab, xlab	labels for the axes. DEFAULT: ""
type, lty, col	arguments passed to plot and lines.
...	further arguments passed to plot like lwd, xaxs, cex.axis, etc.

Details

This function very quickly plots a beta distribution by just specifying alpha and beta.

Value

None. Used for plotting.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2014

See Also

betaPlotComp, normPlot, dbeta, https://cran.r-project.org/package=denstrip, https://cran.r-project.org/view=Distributions

Examples

```
betaPlot()
betaPlot(2,1)
betaPlot(0.5, 2)

# beta distribution is often used for proportions or probabilities
# overview of parameters
# alpha = number of successes + 1.  beta = number of failures + 1
betaPlotComp()
# a bigger: HDI (Highest Density Interval) further to the right (1)
# b bigger: HDI more to the left (0)
# both bigger: narrower HDI, stronger peak
```

---

betaPlotComp *Compare beta distributions*

Description

Visually understand the effect of the beta distribution parameters

Usage

```
betaPlotComp(shape1 = c(0.5, 1:4, 10, 20), shape2 = shape1,
  cumulative = FALSE, cex = 0.8, las = 1, main = "", ylim = lim0(4),
  mar = rep(0, 4), oma = c(2, 2, 4.5, 2), mgp = c(3, 0.7, 0),
  keeppar = FALSE, textargs = NULL, ...)
```

Arguments

shape1	Vector of alpha values as in dbeta. DEFAULT: c(0.5, 1:4, 10,20)
shape2	Beta values to be compared. DEFAULT: shape1
cumulative	Should the cumulative density distribution line be added? DEFAULT: FALSE
cex	Character EXpansion size. DEFAULT: 0.8
las	Label Axis Style passed to axis. DEFAULT: 1
main	Main as in plot. DEFAULT: ""
ylim	LIMit for the Y axis. DEFAULT: lim0(4)
mar	MARgins for plot passed to par. DEFAULT: rep(0,4)
oma	Outer MArgins for plot passed to par. DEFAULT: c(2,2,4.5,2)
mgp	MarGin Placement. DEFAULT: c(3,0.7,0)
keeppar	Should margin parameters be kept instead of being restored to previous value? DEFAULT: FALSE
textargs	List of arguments passed to textField. DEFAULT: NULL
...	Further arguments passed to betaPlot like lines, fill, etc.

Value

None. Used for plotting.

Note

Tries to find suitable subplot for axis labels. This works only for increasing parameter values.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2015

See Also

betaPlot

Examples

```
betaPlotComp()
betaPlotComp(oma=c(2,2,2,2), ylim=lim0(5.5), textargs=list(y=NA))
betaPlotComp(shape1=c(3,10,34), shape2=c(7,9,24))
```

checkFile *check file existance*

Description

check whether files exist and give a useful error/warning/message

Usage

```
checkFile(file, warnonly = FALSE, trace = TRUE)
```

Arguments

file	Filename(s) as character string to be checked for existence.
warnonly	Logical: Only issue a warning instead of an error with stop? DEFAULT: FALSE
trace	Logical: Add function call stack to the message? DEFAULT: TRUE WARNING: in do.call settings with large objects, tracing may take a lot of computing time.

Value

TRUE/FALSE, invisibly

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2016

See Also

file.exists

Examples

```
is.error( checkFile("FileThatDoesntExist.txt")  )
checkFile("FileThatDoesntExist.txt", warnonly=TRUE)
checkFile("FileThatDoesntExist.txt", warnonly=TRUE, trace=FALSE)

## Not run: ## Excluded from CRAN checks because of file creation
# Vectorized:
file.create("DummyFile2.txt")
checkFile(paste0("DummyFile",1:3,".txt"), warnonly=TRUE)
checkFile(paste0("DummyFile",1:3,".txt") )
file.remove("DummyFile2.txt")

compareFiles("dummy.nonexist", "dummy2.nonexist")
checkFile("dummy.nonexist")
```

```
## End(Not run)

dingo <- function(k="brute.nonexist", trace=TRUE)
         checkFile(k, warnonly=TRUE, trace=trace)
dingo()
dingo("dummy.nonexist")

upper <- function(h, ...) dingo(c(h, "dumbo.nonexist"), ...)
upper("dumbo2.nonexist")
upper(paste0("dumbo",2:8,".nonexist"))
upper(paste0("dumbo",2:8,".nonexist"), trace=FALSE)
```

---

ci *calculate confidence interval around mean*

Description

calculate the ends of the confidence interval around mean using `t.test`

Usage

```
ci(dat, lev = 0.95, digits = 3)
```

Arguments

`dat`	Vector with the data to use.
`lev`	Numeric value for confidence level. DEFAULT: 0.95
`digits`	Integer: Number of digits rounded to in output. DEFAULT: 3

Details

Remember that CIs are used when insecurities about the inference from a sample to a general population need quantification, not for hypothesis testing. If two confidence intervals overlap, the difference between the two means still may be significantly different.

Value

A dataframe with the lower and upper confidence interval, as well as the level used.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2010

References

For newbies: Charles Wheelan: naked statistics - stripping the dread from the data, 2013, Norton, ISBN 978-0-393-07195-5.
For statisticians: any of your favorite statistics books should cover confidence intervalls ;-)
http://en.wikipedia.org/wiki/Confidence_interval
Wolfe R, Hanley J (Jan 2002). "If we're so different, why do we keep overlapping? When 1 plus 1 doesn't make 2"
http://www.ecmaj.ca/content/166/1/65.full.pdf
Goldstein, H.; Healey, M.J.R. (1995). "The graphical presentation of a collection of means". Journal of the Royal Statistical Society
http://www.jstor.org/stable/2983411

See Also

t.test

Examples

```
yourdata <- c(5:8,3,14)
ci(yourdata)          # confidence interval with the default confidence level (95%)
ci(yourdata, 0.99)    # specified with a different confidence level
ci(yourdata, 0.99, 4) # returns 4 decimal places
ci(yourdata,,2)       # rounds to 2 decimal places with default level
ci(yourdata)[1,1]     # returns lower boundary of the interval as a numeric
ci(yourdata)[1,2]     # returns upper boundary of the interval as a numeric
ci                    # shows the function itself
```

---

ciBand *polygon confidence bands*

Description

polygon for confidence interval bands, can handle NA's well

Usage

```
ciBand(yu, yl, ym = NULL, x = 1:length(yu), na = "interpolate",
  nastars = TRUE, singlepoints = TRUE, args = NULL, add = FALSE,
  colm = "green3", colb = addAlpha(colm), border = NA, las = 1,
  ylim = range(yu, yl, finite = TRUE), ...)
```

Arguments

yu	y values of upper confidence region boundary
yl	y values of lower confidence region boundary
ym	y values of median/mean line. Only added if this argument is given. DEFAULT: NULL
x	x values (one ascending vector). DEFAULT: 1:length(yu)
na	Method used at NA points. One of "interpolate" or "remove". DEFAULT: "interpolate"
nastars	If na="interpolate", should stars be drawn at places that used to be NA? DEFAULT: TRUE
singlepoints	If na="remove", add points for places surrounded by NAs? can be a boolean (T/F) vector of length three for upper, lower, median. Code to identify isolated points is taken from wq::plotTs. DEFAULT: TRUE
args	List of arguments passed to points for the previous two arguments. DEFAULT: NULL
add	Add to existing plot? If FALSE, plot is called before adding confidence interval. DEFAULT: FALSE
colm	Color for median/mean line. DEFAULT: "green3"
colb	Color of the confidence region band. DEFAULT: addAlpha(colm)
border	polygon border. DEFAULT: NA
las	LabelAxisStyle (axis labels turned upright, see par). DEFAULT: 1
ylim	limits of plot. DEFAULT: range(yu,yl, finite=TRUE)
...	Further arguments passed to plot - or maybe better polygon??

Value

None, currently. Used for drawing.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2015

See Also

quantileBands, polygon, approx2

Examples

```
y1 <- c(1,3,4,2,1,4,6,8,7)
y2 <- c(5,6,5,6,9,8,8,9,10)
y3 <- c(4,4,5,4,4,6,7,8,9)
ciBand(yl=y1, yu=y2, ym=y3)

y1[6:7] <- NA
```

```
ciBand(yl=y1, yu=y2, ym=y3) # interpolation marked with stars if nastars=TRUE
ciBand(yl=y1, yu=y2, ym=y3, na="remove")
lines(y1, col=3, type="o")
lines(y2, col=3, type="o")

y2[1] <- NA
ciBand(yl=y1, yu=y2, ym=y3) # next observation carried backwards (NAs at begin)
# LOCF (last observation carried forwards if NAs at end)
# See ?approx2 for median/mean imputation in these cases
ciBand(yl=y1, yu=y2, ym=y3, na="remove")
y2[9] <- NA
ciBand(yl=y1, yu=y2, ym=y3)
ciBand(yl=y1, yu=y2, ym=y3, na="remove")  # NAs at both ends
y2[1] <- 5
ciBand(yl=y1, yu=y2, ym=y3)
ciBand(yl=y1, yu=y2, ym=y3, na="remove")  # NA only at end

# Actual usefull stuff:  sample size dependency of max and mean
ssdep_max <- function(n) quantile(  replicate(n=200, expr=max(rnorm(n)) )  )
ssdep_mean<- function(n) quantile(  replicate(n=200,expr=mean(rnorm(n)) )  )
x <- 1:100
res_max <- sapply(x, ssdep_max)
res_mean <- sapply(x, ssdep_mean)
ciBand(yl=res_max[2,], yu=res_max[4,], ym=res_max[3,], x=x, ylim=c(-0.5, 3))
ciBand(res_mean[2,], res_mean[4,], res_mean[3,], x=x, add=TRUE, colm="purple")
```

`circle` *Draw circle with a given radius*

Description

Draws a filled circle with a certain radius (in existing plot's units) using `polygon` and `sin`

Usage

```
circle(x, y, r, locnum = 100, ...)
```

Arguments

`x`	x coordinate of points, numeric value of length 1
`y`	y coordinate
`r`	radius of the circle in units of current plot
`locnum`	number of calculated points on the circle (more means smoother but slower). DEFAULT: 100
`...`	further arguments passed to `polygon`, like col, border, lwd

Value

none. Used for drawing.

Note

If circles look like ellipsis, use plot(... asp=1)

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012

See Also

symbols, polygon

Examples

```
plot(1:20, type="n", asp=1)
circle(5,5, r=3)   # 1:1 aspect shows they're really circles and not ellipses.
circle(15,10, r=4, locnum=12, col=2, border=4, lwd=3)

# can not be vectorized:
x <- sample(1:20, 15) ;  y <- sample(1:20, 15) ; r <- runif(20)*3
circle(x,y,r, col=rgb(1,0.5,0,alpha=0.4), border=NA)
for(i in 1:15) circle(x[i],y[i],r[i], col=rgb(1,0.5,0,alpha=0.4), border=NA)
```

---

classify *Classification into groups*

Description

classify continuous values into categories with different methods:
- linearly or logarithmically spaced equal intervals,
- intervals based on quantiles (equally filled bins),
- intervals based on distance from the mean in normal distributions,
- user specified class borders (e.g. for legal or critical limits).

Usage

```
classify(x, method = "equalinterval", breaks, Range = range(x, finite =
  TRUE), sdlab = 1, quiet = FALSE)
```

Arguments

x	Vector with numeric values
method	Character string (partial matching is performed). Classification method or type of binning to compute the class breakpoints. See section Details. DEFAULT: "equalinterval")
breaks	Specification for method, see Details. DEFAULT: different defaults for each method
Range	Ends of color bar for method=equalinterval. DEFAULT: range(x, finite=TRUE)
sdlab	Type of label and breakpoints if `method=standarddeviation`. 1 means `-0.5 sd, 0.5 sd`, 2 means `-1 sd, mean, 1 sd`, 3 means actual numbers for type 1, 4 means numbers for type 2.
quiet	Suppress warnings, eg for values outside Range? DEFAULT: FALSE

Details

Binning methods are explained very nicely in the link in the section References.
nbins indicates the number of classes (and thus, colors).

`method`	explanation	meaning of `breaks`	defau
equalinterval	*nbins* equally spaced classes	nbins	100
quantile	classes have equal number of values	the quantiles (or number of them)	0:4/4
standarddeviation	normal distributions	the number of sd in one direction from the mean	3
logspaced	*nbins* logarithmically spaced	c(nbins, base), see `logSpaced`	c(10
usergiven	custom breakpoints	personal breakpoint values (including ends of Range)	none

The default is set to equalinterval which makes sense for my original intent of plotting lake depth (bathymetry measured at irregularly distributed points) on a linear color scale.
This is the workhorse for `colPoints`.

Value

list with class numbers (index) and other elements for `colPoints`

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2014

References

See this page on the effect of classification (binning) methods:
http://uxblog.idvsolutions.com/2011/10/telling-truth.html

See Also

colPoints

Examples

```
classify( c(1:10, 20), breaks=12)
classify( c(1:10, 20), "q", breaks=0:10/10)
classify( c(1:10, 20), "s", sdlab=2 )
classify( c(1:10, 20), "s", sdlab=1, breaks=2 )
classify( c(1:10, 20), "u", breaks=c(5,27) )
classify( c(1:10, 20), "l")
```

---

climateGraph *climate graph after Walter and Lieth*

Description

Draw a climate diagramm by the standards of Walter and Lieth.

Usage

```
climateGraph(temp, rain, main = "StatName\n52\U{00B0}24' N / 12\U{00B0}58' E\n42 m aSL",
  units = c("\U{00B0}C", "mm"), labs = substr(month.abb, 1, 1), textprop = 0.25,
  ylim = range(temp, rain/2), compress = FALSE, ticklab = -8:30 * 10,
  ticklin = -15:60 * 5, box = TRUE, mar = c(1.5, 2.3, 4.5, 0.2),
  keeppar = TRUE, colrain = "blue", coltemp = "red", lwd = 2,
  arghumi = NULL, argarid = NULL, argcomp = NULL, arggrid = NULL,
  argtext = NULL, ...)
```

Arguments

temp	monthly temperature mean in degrees C
rain	monthly rain sum in mm (12 values)
main	location info as character string. can have \n. DEFAULT: "StatName\n52d 24' N / 12d 58' E\n42 m aSL"
units	units used for labelling. DEFAULT: c("d C", "mm")
labs	labels for x axis. DEFAULT: J,F,M,A,M,J,J,A,S,O,N,D
textprop	proportion of graphic that is used for writing the values in a table to the right. DEFAULT: 0.25
ylim	limit for y axis in temp units. DEFAULT: range(temp, rain/2)
compress	should rain>100 mm be compressed with adjusted labelling? (not recommended for casual visualization!). DEFAULT: FALSE
ticklab	positions for vertical labelling. DEFAULT: -8:30*10

ticklin	positions for horizontal line drawing. DEFAULT: -15:60*5
box	draw box along outer margins of graph? DEFAULT: TRUE
mar	plot margins. DEFAULT: c(1.5,2.3,4.5,0.2)
keeppar	Keep the changed graphical parameters? DEFAULT: TRUE
colrain	Color for rain line and axis labels. DEFAULT: "blue"
coltemp	color for temperature line and axis labels. DEFAULT: "red"
lwd	line width of actual temp and rain lines. DEFAULT: 2
arghumi	List of arguments for humid polygon, like density, angle. DEFAULT: NULL (internal x,y, col, border)
argarid	List of arguments for arid area. DEFAULT: NULL
argcomp	List of arguments for compressed rainfall polygon. DEFAULT: NULL
arggrid	List of arguments for bachground grid lines. DEFAULT: NULL
argtext	List of arguments for text at right hand if textprop>0. DEFAULT: NULL
...	further arguments passed to plot, like col.main

Value

None. Plots data and table.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, June 2013

References

Heinrich Walter, Helmut Lieth: Klimadiagramm-Weltatlas. Gustav Fischer Verlag, Jena 1967
Examples:
http://www.hoelzel.at/_verlag/geojournal/archiv/klima/2006_01/lieth.gif
http://www.hoelzel.at/_verlag/geojournal/archiv/klima/istanbul/istanbul200.gif
http://www.ipb.uni-tuebingen.de/kurs/comp/1_excel2007/1_pic/2007diagramm_verbund02.jpg
http://www.zivatar.hu/felhotar/albums/userpics/wldp.png

See Also

diagwl in package climatol

Examples

```
temp <- c(-9.3,-8.2,-2.8,6.3,13.4,16.8,18.4,17,11.7,5.6,-1,-5.9)#
rain <- c(46,46,36,30,31,21,26,57,76,85,59,46)

climateGraph(temp, rain)
climateGraph(temp, rain, textprop=0.6)
climateGraph(temp, rain, mar=c(2,3,4,3), textprop=0) # no table written to the right
# vertical lines instead of filled polygon:
```

```
climateGraph(temp, rain, arghumi=list(density=15, angle=90))
# fill color for arid without transparency:
climateGraph(temp, rain, argarid=list(col="gold"))
# for the Americans - axes should be different, though!:
climateGraph(temp, rain, units=c("\U{00B0}F","in"))

rain2 <- c(23, 11, 4, 2, 10, 53, 40, 15, 21, 25, 29, 22)
# fix ylim if you want to compare diagrams of different stations:
climateGraph(temp, rain2, ylim=c(-15, 50)) # works with two arid phases as well

op <- par(mfrow=c(2,1)) # mulipanel plot
climateGraph(temp, rain, argtext=list(cex=0.7))
climateGraph(temp, rain2, argtext=list(cex=0.7))
par(op)

rain <- c(54, 23, 5, 2, 5, 70, 181, 345, 265, 145, 105, 80) # with extrema
climateGraph(temp, rain) # August can be visually compared to June
climateGraph(temp, rain, compress=TRUE)
# compressing extrema enables a better view of the temperature,
# but heigths of rain cannot be visually compared anymore
climateGraph(temp, rain, compress=TRUE, ylim=c(-10, 90))
# needs ylim in linearly continued temp units
climateGraph(temp, rain, compress=TRUE, argcomp=list(density=30, col=6))

## Not run:
pdf("ClimateGraph.pdf")
climateGraph(temp, rain, main="Another Station\nlocated somewhere else")
dev.off()
system2("open", "ClimateGraph.pdf")
unlink("ClimateGraph.pdf")

# further German reading:
browseURL("http://www.klimadiagramme.de/all.html")


# Climate Graphs for the USA:
NOOAlink <- "http://www1.ncdc.noaa.gov/pub/data/normals/1981-2010/"
browseURL(NOOAlink)
# Find your Station here:
browseURL(paste0(NOOAlink,"/station-inventories/allstations.txt"))

# Data from Roseburg, Oregon:
download.file(destfile="Roseburg.txt", url=paste0("http://www1.ncdc.noaa.gov/",
          "pub/data/normals/1981-2010/products/station/USC00357331.normals.txt"))
RT <- read.table(file="Roseburg.txt", skip=11, nrows=1, as.is=TRUE)[1,-1]
RT <- ( as.numeric(substr(RT,1,3))/10   - 32) * 5/9     # converted to degrees C
RP <- read.table(file="Roseburg.txt", skip=580, nrows=1, as.is=TRUE)[1,-1]
RP <-  as.numeric(substr(RP,1,nchar(RP)-1))/100*25.4
meta <- read.table(file="Roseburg.txt", nrows=5, as.is=TRUE, sep=":")
meta <- paste(meta[1,2], paste(meta[3:4 ,2], collapse=" /"), meta[5,2], sep="\n")
unlink("Roseburg.txt")

climateGraph(RT, RP, main=meta)
```

```
climateGraph(RT, RP, main=meta, compress=TRUE)


# Climate Graphs for Germany:
browseURL("https://github.com/brry/rdwd#rdwd")
instGit("brry/rdwd")
link <- rdwd::selectDWD("Potsdam", res="monthly", var="kl", per="h")
clim <- rdwd::dataDWD(link, dir=tempdir())
dates <- strptime(clim$MESS_DATUM_BEGINN, "%Y%m%d")
temp <- tapply(clim$LUFTTEMPERATUR, INDEX=format(dates, "%m"), FUN=mean, na.rm=FALSE)
precsums <- tapply(clim$NIEDERSCHLAGSHOEHE, INDEX=format(dates, "%Y-%m"), FUN=sum)
eachmonth <- format(strptime(paste(names(precsums),"01"), "%Y-%m %d"),"%m")
prec <- tapply(precsums, eachmonth, FUN=mean)
meta <- paste("Potsdam\n", paste(range(dates, na.rm=TRUE), collapse=" to "), "\n", sep="")

climateGraph(temp, prec, main=meta, ylim=c(-2, 45))
# Add Quartiles (as in boxplots): numerically sorted, 50% of the data lie inbetween
TQ <- tapply(clim$LUFTTEMPERATUR, INDEX=format(dates, "%m"), FUN=quantile)
TQ <- sapply(TQ, I)
arrows(x0=1:12, y0=TQ["25%",], y1=TQ["75%",], angle=90, code=3, col=2, len=0.1)
#
PQ <- tapply(precsums, eachmonth, FUN=quantile)
PQ <- sapply(PQ, I)
arrows(x0=1:12, y0=PQ["25%",]/2, y1=PQ["75%",]/2, angle=90, code=3, col=4, len=0, lwd=3, lend=1)
mtext("IQR shown als lines", col=8, at=6.5, line=0.7, cex=1.2, font=2)


# Comparison to diagram in climatol
install.packages("climatol")
help(package="climatol")
library(climatol)
data(datcli)
diagwl(datcli,est="Example station",alt=100,per="1961-90",mlab="en")


## End(Not run)
```

---

colPoints *Points colored relative to third dimension*

Description

Draw colored points for 3D-data in a 2D-plane. Color is relative to third dimension, by different classification methods. Can take 3 vectors or, as in image, 2 vectors and a matrix for z.

Usage

```
colPoints(x, y, z, data, add = TRUE, col = seqPal(cl$nbins), col2 = c(NA,
  "grey", "black"), Range = range(z, finite = TRUE),
```

```
method = "equalinterval", breaks, sdlab = 1, legend = TRUE,
legargs = NULL, hist = FALSE, histargs = NULL, lines = FALSE,
nint = 30, xlab = deparse(substitute(x)), ylab = deparse(substitute(y)),
zlab = deparse(substitute(z)), axes = TRUE, las = 1, pch = 16,
x1 = 0.6, y1 = 0.88, x2 = 0.99, y2 = 0.99, density = NULL,
quiet = FALSE, ...)
```

Arguments

x, y	Vectors with coordinates of the points to be drawn
z	z values belonging to coordinates. Vector or matrix with the color-defining height values
data	Optional: data.frame with the column names as given by x,y and z.
add	Logical. Should the points be added to current (existing!) plot? If FALSE, a new plot is started. DEFAULT: TRUE (It's called col**Points**, after all)
col	Vector of colors to be used. DEFAULT: 100 colors from sequential palette seqPal (color-blind safe, black/white-print safe)
col2	Color for points where z is NA, or lower / higher than Range. DEFAULT: c(NA, 1, 8)
Range	Ends of color bar. If NULL, it is again the DEFAULT: range(z, finite=TRUE)
method	Classification method (partial matching is performed), see classify (ways to get color breakpoints). DEFAULT: "equalinterval")
breaks	Specification for method, see classify. DEFAULT: different defaults for each method
sdlab	Type of label and breakpoints if method=standarddeviation, see classify. DEFAULT: 1
legend	Logical. Should a colPointsLegend be drawn? DEFAULT: TRUE
legargs	List. Arguments passed to colPointsLegend. DEFAULT: NULL, with some defaults specified internally
hist	Logical. Should a colPointsHist be drawn? DEFAULT: FALSE (TRUE if histargs are given)
histargs	List. Arguments passed to colPointsHist. DEFAULT: NULL
lines	Logical. Should lines be drawn instead of / underneath the points? (color of each segments is taken from starting point, last point is endpoint.) If lines=TRUE and pch is not given, pch ist set to NA. DEFAULT: FALSE
nint	Numeric of length 1. Number of interpolation points between each coordinate if lines=TRUE. nint=1 means no interpolation. Values below 10 will smooth coordinates and might miss the original points. DEFAULT: 30
xlab	x-axis label. DEFAULT: deparse(substitute(x))
ylab	y-axis label. DEFAULT: ditto
zlab	colPointsLegend title. DEFAULT: ditto
axes, las	Draw axes? Label Axis Style. Only used when add=FALSE. See par. DEFAULT: axes=TRUE, las=1 (all labels horizontal)

pch	Point CHaracter. See par. DEFAULT: 16
x1, x2, y1, y2	Relative coordinates [0:1] of inset plot, see smallPlot. Passed to colPointsLegend. DEFAULT: x: 0.6-0.99, y: 0.88-0.98
density	Arguments for density line in colPointsLegend DEFAULT: NULL
quiet	Turn off warnings? DEFAULT: FALSE
...	Further graphical arguments passed to plot, points and segments, eg cex, xlim (when add=F), mgp, main, sub, asp (when add=F), etc. Note: col does not work, as it is already another argument

Value

Invisible list of values that can be passed to colPointsLegend or colPointsHist.

Note

Rstudio scales graphics really badly, so don't expect the right legend width out of the box if you use Rstudio! Exporting via `png("myplot.png", 600,400); colPoints(x,y,z); dev.off()` usually works much better

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011-2014. I'd be interested in hearing what you used the function for.

References

http://uxblog.idvsolutions.com/2011/10/telling-truth.html, http://www.theusrus.de/blog/the-good-the-bad-22012/

See Also

classify, colPointsLegend, colPointsHist

Examples

```
i <- c( 22,  40,  48,  60,  80,  70,  70,  63,  55,  48,  45,  40,  30,  32)
j <- c(  5,  10,  15,  20,  12,  30,  45,  40,  30,  36,  56,  33,  45,  23)
k <- c(175, 168, 163, 132, 120, 117, 110, 130, 131, 160, 105, 174, 190, 183)

# basic usage:
colPoints(i,j,k, cex=1.5, pch="+", add=FALSE)

# with custom Range:
colPoints(i,j,k, cex=1.5, pch="+", add=FALSE, Range=c(150,190), density=FALSE)
# can be used to allow comparison between several plots
# points outside the range are plotted with col2

# with custom colors:
mycols <- colorRampPalette(c("blue","yellow","red"))(50)
```

```
colPoints(i,j,k, cex=1.5, pch="+", add=FALSE, col=mycols)

# With legend title:
colPoints(i,j,k, cex=2, add=FALSE, zlab="Elevation [m above NN.]",
         legargs=list(density=FALSE))
?colPointsLegend # to see which arguments can be set via legargs

# with lines (nint to change number of linear interpolation points):
colPoints(i,j,k, cex=1.5, add=FALSE, lines=TRUE, nint=10, lwd=2)
# With NAs separating lines:
tfile <- system.file("extdata/rivers.txt", package="berryFunctions")
rivers <- read.table(tfile, header=TRUE, dec=",")
colPoints(x,y,n, data=rivers, add=FALSE, lines=TRUE)
colPoints(x,y,n, data=rivers, add=FALSE, lines=TRUE, pch=3)
colPoints(x,y,n, data=rivers, add=FALSE, lines=TRUE, pch=3, nint=2)

# different classification methods:
set.seed(007) ;  rx <- rnorm(30) ; ry <- rnorm(30) ; rz <- rnorm(30)*100
# sd: normal distribution
mycols <- colorRampPalette(c("blue","yellow", "red"))
colPoints(rx,ry,rz, add=FALSE, col=mycols(5), method="s",
          legargs=list(horiz=FALSE, x1=0.7, x2=0.95))
colPoints(rx,ry,rz, add=FALSE, col=mycols(6), method="s", sdlab=2,
          legargs=list(horiz=FALSE, labelpos=5, lines=FALSE, title=""))
# quantiles: each color is equally often used
colPoints(rx,ry,rz, add=FALSE, method="q",
          legargs=list(mar=c(0,1,1,0), bg="transparent") )
text(rx,ry,round(rz), col=8)
# logSpaced for rightly skewed data:
set.seed(41); rz2 <- rbeta(30, 1,7)*100
colPoints(rx,ry,rz2, add=FALSE, method="l", breaks=c(20,1.1708), col=mycols(20))
colPoints(rx,ry,rz2, add=FALSE, method="q", breaks=0:20/20, col=mycols(20),
          legargs=list(at=pretty2(rz2), labels=pretty2(rz2)) )

# With histogram:
colPoints(i,j,k, add=FALSE, hist=TRUE)
colPoints(i,j,k, cex=3.5, lwd=3, pch=1, histargs=list(bg=5, breaks=5), add=FALSE)
colPoints(rx,ry,rz, cex=3.5, lwd=3, pch=1, add=FALSE, legend=FALSE,
   histargs=list(mar=c(0,0,0,0), x1=0.5, x2=1, y1=0.8, y2=0.99, yaxt="n"))

# use classify separately:
text(rx,ry,round(rz), col=mycols(100)[classify(rz)$index], cex=0.7)

# histogram in lower panel:
layout(matrix(1:2), heights=c(8,4) )
colPoints(i,j,k, add=FALSE, y1=0.8, y2=1)
colPointsHist(z=k, x1=0.05, x2=1, y1=0, y2=0.4, mar=3, outer=TRUE)
layout(1)


# Customizing the legend :
cp <- colPoints(i,j,k, legend=FALSE, add=FALSE)
colPointsLegend(x1=0.2, x2=0.95, y1=0.50, y2=0.40, z=k, labelpos=5, atminmax=TRUE, bg=7)
```

```
colPointsLegend(x1=0.5, x2=0.90, y1=0.28, y2=0.18, z=k, Range=c(80, 200), nbins=12, font=3)
colPointsLegend(x1=0.1, x2=0.40, y1=0.15, y2=0.05, z=k, labelpos=5, lines=FALSE, title="")
colPointsLegend(z=k, horizontal=FALSE)
colPointsLegend(x1=0.01, y2=0.80, z=k, horizontal=FALSE, labelpos=4, cex=1.2)
colPointsLegend(x1=0.23, y2=0.95, z=k, horizontal=FALSE, labelpos=5, cex=0.8,
  dens=FALSE, title="", at=c(130,150,170), labels=c("y","rr","Be"), lines=FALSE)
# For method other than colPoints' default, it is easiest to include these
# options as a list in legargs, but you can also use the invisible output
# from colPoints for later calls to colPointsLegend
do.call(colPointsLegend, cp)
do.call(colPointsLegend, owa(cp, list(colors=rainbow2(100), cex=1.2)))

# colPoints with matrix:
colPoints(z=volcano, add=FALSE)
# image and contour by default transpose the matrix!
# colPoints shows what is really in the data.
colPointsHist(z=volcano)

# highlight local character of points on a regular grid normally drawn with image:
z <- t(volcano)  ;  x <- 1:ncol(z)  ;  y <- 1:nrow(z)
colPoints(x,y,z, add=FALSE)  # takes matrix for z
contour(x,y,t(z), add=TRUE)

# image only takes a regular matrix, but not scatterpoints...
image(x,y,t(z), col=rainbow2(100))

# add single newly measured points to image (fictional data):
mx <- c( 22,  40,  80,  45,  60,  63,  30,  70)
my <- c(  5,  33,  12,  56,  20,  40,  45,  45)
mz <- c(135, 155, 120, 105, 140, 130, 190, 110)
colPoints(mx,my,mz, cex=5, pch="*", Range=c(94, 195), col=rainbow2(100), col2=NA, legend=FALSE)
points(mx,my, cex=4)
text(mx,my,mz, adj=-0.5, font=2)

# santiago.begueria.es/2010/10/generating-spatially-correlated-random-fields-with-r
if(require(gstat)){
xyz <- gstat(formula=z~1, locations=~x+y, dummy=TRUE, beta=1,
             model=vgm(psill=0.025,model="Exp",range=5), nmax=20)
xyz <- predict(xyz, newdata=data.frame(x=runif(200, 20,40),y=runif(200, 50,70)), nsim=1)
head(xyz)
colPoints(x,y,sim1, data=xyz, col=rainbow2(100), add=FALSE)
}
```

`colPointsHist` *Histogram for colPoints*

Description

Adds Histogram to plots created or enhanced with `colPoints`

Usage

```
colPointsHist(z, nbins = 40, colors = seqPal(nbins), bb = seqR(z,
  length.out = nbins + 1), at = pretty2(z), labels = at, bg = "white",
  x1 = 0, x2 = 0.4, y1 = 0, y2 = 0.3, outer = FALSE, mar = c(2, 2,
  1, 0.5), mgp = c(1.8, 0.6, 0), sborder = NA, resetfocus = TRUE,
  breaks = 20, freq = TRUE, col = par("fg"), border = NA, main = "",
  ylab = "", xlab = "", las = 1, axes = TRUE, ...)
```

Arguments

z	Values of third dimension used in colPoints
nbins	Number of classes (thus, colors). DEFAULT: 40
colors	Colors that are used for the background. DEFAULT: seqPal(nbins)
bb	Borders of bins for the background. DEFAULT: seqR(z, length.out=nbins+1)
at	Positions of x-axis labels. DEFAULT: pretty2(z)
labels	X-axis labels themselves. DEFAULT: at
bg	Background behind background and axis labels. DEFAULT: "white"
x1, x2, y1, y2	Relative coordinates [0:1] of inset plot, see smallPlot. DEFAULT: x: 0-0.3, y: 0-0.4
outer	Logical: Should legend be relative to device instead of current figure? use outer=TRUE when par(mfrow, oma) is set. DEFAULT: FALSE
mar	Margins for smallPlot. DEFAULT: c(2, 2, 1, 0.5)
mgp	MarGinPlacement: distance of xlab/ylab, numbers and line from plot margin, as in par, but with different defaults. DEFAULT: c(1.8, 0.6, 0)
sborder	Border around inset subplot. DEFAULT: par("fg")
resetfocus	Reset focus to original plot? Specifies where further low level plot commands are directed to. DEFAULT: TRUE
breaks	Breaks as in hist, but with a different default. DEFAULT: 20
freq	Plot count data in hist? (if FALSE, plot density instead). DEFAULT: TRUE
col	Color of histogram bars. DEFAULT: par("fg")
border	Border around each bar. DEFAULT: NA
main, ylab, xlab	Labels. DEFAULT: ""
las	LabelAxisStyle. DEFAULT: 1
axes	Draw axes?. DEFAULT: TRUE
...	Further arguments passed to hist. NOT POSSIBLE: x, add

Value

invisible list of par of smallPlot, adds histogram to current plot

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Aug 2014

See Also

colPointsLegend and colPoints for real life examples

Examples

```
z <- rnorm(50)
plot(1:10)
colPointsHist(z=z)
```

---

colPointsLegend *Legend for colPoints*

Description

Adds legends to plots created or enhanced with colPoints

Usage

```
colPointsLegend(z, Range = range(z, finite = TRUE), nbins = 100,
  colors = seqPal(nbins), bb = seqR(Range, length.out = nbins + 1),
  at = pretty2(Range), labels = at, adj = 0.5, x1 = 0.6, y1 = 0.88,
  x2 = 0.99, y2 = 0.99, outer = FALSE, xpd = NA, mar, mgp = c(1.8,
  0.6, 0), bg = "white", sborder = NA, resetfocus = TRUE,
  plottriangle = FALSE, triangle = 0.14, tricol = c(8, 1),
  density = NULL, lines = TRUE, atminmax = FALSE, horizontal = TRUE,
  labelpos = 1, titlepos = 3, title = "Legend", las = 1, x, y, index,
  ...)
```

Arguments

z	Values of third dimension used in colPoints, can be a matrix or a vector etc, but must be numeric
Range	Ends of color bar for method=equalinterval. DEFAULT: range(z, finite=TRUE)
nbins	Number of classes (thus, colors). If colors is given, nbins is overwritten with length(colors). DEFAULT: 100
colors	Color vector. DEFAULT: seqPal from yellow (lowest) to blue (highest value in Range)
bb	Borders of bins for the legend (key). DEFAULT: seqR(Range, length.out=nbins+1)
at	Positions of legend labels. DEFAULT: pretty2(Range)
labels	Labels that are written at the positions of at. DEFAULT: at

adj	label adjustment parallel to legend bar (only one number!). DEFAULT: 0.5
x1, x2, y1, y2	Relative coordinates [0:1] of inset plot, see smallPlot. DEFAULT: x: 0.6-0.99, y: 0.88-0.99
outer	Logical: Should legend be relative to device instead of current figure? use outer=TRUE when par(mfrow, oma) is set. DEFAULT: FALSE
xpd	Logical: should text be expanded outisde of plotting region? Must be NA if outer=TRUE. DEFAULT: NA
mar	Margins for smallPlot. DEFAULT: internal calculations based on title, labelpos and titlepos.
mgp	MarGinPlacement: distance of xlab/ylab, numbers and line from plot margin, as in par, but with different defaults. DEFAULT: c(1.8, 0.6, 0)
bg	Background behind key, labels and title. DEFAULT: "white"
sborder	Border around inset subplot. DEFAULT: NA
resetfocus	Reset focus to original plot? Specifies where further low level plot commands are directed to. DEFAULT: TRUE
plottriangle	Should triangles be plotted at the end of the legend for values outside Range? Vector of length two (for lower and upper, internally recycled). If this argument is missing but triangle is given, this is set to TRUE. DEFAULT: FALSE
triangle	Percentage of bar length at lower and upper end for triangles (can be a vector with two different values). DEFAULT: 0.14
tricol	Triangle colors for lower and upper end. DEFAULT: c(8,1)
density	List of arguments passed to kernel density estimation. Can also be FALSE to suppress KDE line drawing. DEFAULT: NULL
lines	Plot black lines in the color bar at at? DEFAULT: TRUE
atminmax	Should the extrema of the legend be added to at? DEFAULT: FALSE
horizontal	Horizontal bar? if FALSE, a vertical bar is drawn. DEFAULT: TRUE
labelpos	Position of labels relative to the bar. Possible: 1 (below), 2 (left), 3 (above), 4 (right), 5(on top of bar). DEFAULT: 1
titlepos	Position of title -"-. DEFAULT: 3
title	Legend title. DEFAULT: "Legend"
las	LabelAxisStyle. DEFAULT: 1
x, y, index	Ignored arguments, so that you can pass the result from colPoints via do.call(colPointsLegend, cp_
...	Further arguments passed to text and strwidth, e.g. cex, srt, font, col. But NOT adj!

Value

invisible list of par of smallPlot, adds legend bar to current plot

Note

x1,x2,y1,y2,labelpos,titlepos,title have different defaults when horizontal=FALSE

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012-2014

See Also

colPointsHist, colPoints for real life example

Examples

```
z <- rnorm(50)
plot(1:10)
colPointsLegend(z=z)
colPointsLegend(z=z, titlepos=2)
colPointsLegend(z=z, horiz=FALSE) # note the different defaults
# positioning relative to plot:
colPointsLegend(z=z, x1=0.05, x2=0.3, y1=0.7,y2=0.9, title="Booh!", density=FALSE)
# Denote values outside of Range wit a triangle:
colPointsLegend(z=z, Range=c(-1,3), x1=0.2, y1=0.4, y2=0.6, triangle=0.2)
colPointsLegend(z=z, horiz=FALSE, x1=0.7, y1=0.6, plottriangle=TRUE, density=FALSE)
?colPoints # example section for actual usage
```

---

combineFiles *Combine Textfiles into one*

Description

Combine several textfiles into one, regardless of their content.

Usage

```
combineFiles(inFiles = dir(), inDir = getwd(),
  outFile = "combined_Textfiles.txt", outDir = inDir, sep = NULL,
  names = TRUE, selection = NULL, progbar = !quiet, quiet = FALSE)
```

Arguments

inFiles	vector with names of input files, as can be read with scan. Is pasted with inDir, so don't use full paths. DEFAULT: dir()
inDir	Character string: path to the files. E.g. "D:/MyFolder/Subfolder". Don't have / at the end. DEFAULT: getwd().
outFile	Character string: name of the file to be created. Again, just the file name, not a path. DEFAULT: "combined_Textfiles.txt"
outDir	Character string: path for output file. DEFAULT: inDir

sep	Character string: Separation between content of each file and the following. DEFAULT: NULL, with which it uses an empty line, two lines with dashes, and another line break.
names	Should File names be included after sep? DEFAULT: TRUE
selection	Index of rows that should be written. Can refer to each file separately, e.g. `substr(inFile_i,1,1)=="#"`, DEFAULT: all lines
progbar	Should a progress bar be drawn? Useful if you combine many large files. DEFAULT: !quiet, i.e. TRUE
quiet	Suppress message about number of files combined? DEFAULT: FALSE

Value

None, but prints number of files combined and output file name.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Nov 2012, Dec 2014, Jul 2015

See Also

compareFiles, and the functions used internally here, namely: paste, scan, write.

Examples

```
## These are skipped by rcmd check (writing to external places is not allowed)
## Not run:
cat("This is Sparta.\nKicking your face.", file="BujakashaBerry1.txt")
cat("Chuck Norris will roundhousekick you.", file="BujakashaBerry2.txt")
combineFiles(inFiles=paste0("BujakashaBerry", 1:2, ".txt"),
             outFile="BujakashaBerry3.txt")
file.show("BujakashaBerry3.txt")
unlink(paste0("BujakashaBerry", 1:3, ".txt"))

## End(Not run)
```

---

compareFiles *Compare textfiles for equality*

Description

Returns the line numbers where two (text)files differ

Usage

```
compareFiles(file1, file2, nr = 20, startline = 1, endline = length(f1),
  quiet = FALSE, ...)
```

Arguments

file1, file2	Filenames to be read by readLines.
nr	number of results printed. DEFAULT: 20
startline, endline	start and end lines, e.g. to exclude section that is already compared.
quiet	show warnings about file lengths? DEFAULT: FALSE
...	further arguments passed to readLines

Value

Vector of line numbers that differ, result from head(..., nr)

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Aug 2014

See Also

http://text-compare.com/ which I sadly only discovered after writing this function, dupes for finding duplicate lines, combineFiles

Examples

```
filenames <- system.file(paste0("extdata/versuch",1:2,".txt"), package="berryFunctions")
compareFiles(filenames[1], filenames[2], warn=FALSE)
```

---

convertUmlaut *Convert German Umlaute to ASCII*

Description

Convert German Umlaute (ae, oe, ue, ss) to ASCII. Conversion happens case sensitive for the first three.

Usage

```
convertUmlaut(x)
```

Arguments

x	Character string(s) containing German Umlaute

Value

Character strings

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Oct-Nov 2016

See Also

tools::showNonASCII, gsub, iconv(x, to="ASCII//TRANSLIT")

Examples

```
## Not run:
link <- paste0("ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/",
               "monthly/kl/recent/KL_Monatswerte_Beschreibung_Stationen.txt")
weatherstations <- read.fwf(link, widths=c(6,9,10,16,11,8,41,99), skip=3)
examples <- removeSpace(weatherstations[c(153, 509, 587, 2, 651, 851),7])
examples
convertUmlaut(examples) # note how lower and upper case is kept

## End(Not run)
```

---

createFun *create function framework*

Description

create a file with a complete (Roxygen) framework for a new function in this package

Usage

```
createFun(fun, package = "berryFunctions", path = "S:/Dropbox/Rpack")
```

Arguments

fun	Character string or unquoted name. Function that will be crated with identical filename
package	Character String with package name. DEFAULT: "berryFunctions"
path	Path to package in development (not package name itself) DEFAULT: "S:/Dropbox/Rpack"

Details

Tries to open the file in the standard editor for .R files using system2

Value

file name as character string

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, March 2016

See Also

system2, funSource, Roxygen2: https://cran.r-project.org/package=roxygen2/vignettes/rd.html

Examples

```
#createFun("myNewFunction")
```

---

createPres *Create .Rnw presentation template*

Description

Create folder with .Rnw presentation template and fig_extern folder.

Usage

```
createPres(presname = "presentation", dir = "presentation", path = ".")
```

Arguments

presname	Name of .Rnw file to be created. DEFAULT: "presentation"
dir	Name of directory that will contain .Rnw file and fig_extern folder. "_1" will be appended if already existing, see newFilename. DEFAULT: "presentation"
path	Location of dir. Passed to setwd. DEFAULT: "."

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Mar 2017

See Also

createFun

Examples

```
createPres("Berry_Conference")
```

dataDWD *dataDWD and readDWD are now in 'rdwd'.*

Description

dataDWD and readDWD have been greatly improved and are now in the package rdwd. This provides a much better infrastructure for file selection and downloading. See https://github.com/brry/rdwd#rdwd for usage instructions.

dataStr *str of datasets*

Description

Print the str of each dataset returned by data, by default in the package datasets

Usage

```
dataStr(package = "datasets", ...)
```

Arguments

package	package. DEFAULT: "datasets"
...	other arguments passed to data

Value

NULL. prints via message in a for loop.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, November 2015, in search of good datasets for teaching

See Also

str

Examples

```
# dataStr()
```

distance *Distance between points*

Description

Calculate distance between points on planar surface

Usage

```
distance(x, y, xref, yref, along = FALSE)
```

Arguments

x	vector with x-coordinate(s) of point(s)
y	ditto for y
xref	single x coordinate of reference point
yref	ditto for y
along	Logical: Should distances be computed along vector (x,y)? If TRUE, (xref,yref) are ignored. If both (xref,yref) are not given, along is set to TRUE.

Details

The function is quite simple: `sqrt((xref - x)^2 + (yref - y)^2)`

Value

vector with the distances

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012

See Also

nndist in the package spatstat for distance to nearest neighbour

Examples

```
A <- c(3,  9,-1)
B <- c(7, -2, 4)
plot(A,B)
text(A,B, paste0("P",1:3), adj=1.1)
points(3,5, col=2, pch=16)
segments(3,5, A,B)
distance(A,B, 3,5)
text(c(3.2,6,1), c(6,1,4), round(distance(A,B, 3,5),2) )
```

divPal *Diverging color palette*

Description

Diverging color palette: brown to blue, light colors in the middle, darker at the extremes, good for displaying values in two directions

Usage

```
divPal(n = 12, reverse = FALSE, alpha = 1, rwb = FALSE, ryb = FALSE,
  colors = NULL, ...)
```

Arguments

n	Number of colors. DEFAULT: 12
reverse	Reverse colors? DEFAULT: FALSE
alpha	Transparency (0=transparent, 1=fully colored). DEFAULT: 1
rwb	Should colors be in red-white-blue instead of brown-blue? DEFAULT: FALSE
ryb	Use red-yellow-blue instead of the default, with "khaki" in the center. DEFAULT: FALSE
colors	If not NULL, a color vector used in colorRampPalette. DEFAULT: NULL
...	Further arguments passed to colorRamp

Value

Character string vector with color names

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2016

References

Originally in 12 shades in the IPCC Assesment Report 5 Chapter 12 Fig 12.22, http://www.ipcc.ch/report/ar5/wg1/

See Also

showPal, seqPal, addAlpha, colorRampPalette, package RColorBrewer

Examples

```
plot(rep(1,12), pch=16, cex=5, col=divPal(12), xaxt="n")
showPal()
```

dupes *Duplicate lines in file*

Description

Number of duplicates per line of (text) file. Per default saved to file which can be loaded into excel / libreoffice. With conditional formatting of the first column, colors show for each line how often it occurs in the file. A LibreOffice file is included. Note: OpenOffice does not provide color scales based on cell values.

Usage

```
dupes(file, ignore.empty = TRUE, ignore.space = TRUE, tofile = missing(n),
  n = length(d))
```

Arguments

`file`	File name (character string)
`ignore.empty`	Should empty lines be ignored? DEFAULT: TRUE
`ignore.space`	Should leading/trailing whitespace be ignored? DEFAULT: TRUE
`tofile`	Logical: should output be directed to a file? Otherwise, a dataframe with line numbers and number of duplicates of that line will be printed in the console. DEFAULT: missing(n)
`n`	Show only the first n values if `tofile=FALSE`. DEFAULT: length(d)

Value

Either: a data.frame with line numbers of duplicate rows and the number of duplicates
Or: a file is written with the number of duplicates and the original `file` content.

Note

This has not been tested al that much - feedback is heavily welcome!

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2014

See Also

compareFiles

Examples

```
file <- system.file("extdata/doublelines.txt", package="berryFunctions")
dupes(file, tofile=FALSE)
dupes(file, tofile=FALSE, ignore.empty=TRUE)

## These are skipped by rcmd check (opening external places is not allowed):
## Not run: dupes(file)

# a template file (dupes.ods) for libreOffice Calc is available here:
system.file("extdata", package="berryFunctions")

## Not run: system2("nautilus", system.file("extdata/dupes.ods", package="berryFunctions"))

# To open folders with system2:
# "nautilus" on linux ubuntu
# "open" or "dolphin" on mac
# "explorer" or "start" on windows
```

---

exp4p *4-parametric exponential function*

Description

Fits an exponential function of the form a*e^(b*(x+c))+d

Usage

```
exp4p(x, y, digits = 2, plot = FALSE, las = 1, col = 1:6,
  legarg = NULL, ...)
```

Arguments

x, y	x and y Data
digits	significant digits for rounding R^2. DEFAULT: 2
plot	plot data and fitted functions? DEFAULT: FALSE
las	label axis style, see par. DEFAULT: 1
col	6 colors for lines and legend texts. DEFAULT: 1:6
legarg	Arguments passed to legend. DEFAULT: NULL
...	further graphical parameters passed to plot

Details

This is mainly a building block for mReg

Value

Data.frame with the 4 parameters for each optim method

Note

Optim can be slow! It refers to the functions rmse and rsquare, also in this package. L-BFGS-B needs finite values. In case it doesn't get any with the initial parameters (as in the first example Dataset), it trys again with the parameters optimized via Nelder Mead.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012-2013, outsourced from mReg in July 2014

See Also

mReg, lm

Examples

```
# exponential decline of temperature of a mug of hot chocolate
tfile <- system.file("extdata/Temp.txt", package="berryFunctions")
temp <- read.table(tfile, header=TRUE, dec=",")
head(temp)
plot(temp)
temp <- temp[-20,] # missing value - rmse would complain about it
x <- temp$Minuten
y <- temp$Temp
rm(tfile, temp)

exp4p(x,y, plot=TRUE)
# y=49*e^(-0.031*(x - 0  )) + 25 correct, judged from the model:
# Temp=T0 - Te *exp(k*t) + Te     with    T0=73.76,  Tend=26.21, k=-0.031
# optmethod="Nelder-Mead"  # y=52*e^(-0.031*(x + 3.4)) + 26 wrong
```

---

expReg *Exponential regression with plotting*

Description

uses lm; plots data if add=FALSE, draws the regression line with abline and confidence interval with polygon and writes the formula with legend

Usage

```
expReg(x, y = NULL, data = NULL, logy = TRUE, predictnew = NULL,
  interval = "confidence", plot = TRUE, digits = 2, inset = 0,
  xpd = par("xpd"), pos1 = "top", pos2 = NULL, add = FALSE, pch = 16,
  col = rgb(0, 0, 0, 0.5), modcol = 2, lwd = 1,
  xlab = deparse(substitute(x)), ylab = deparse(substitute(y)),
  main = "exponential regression", xlim = range(x), ylim = range(y), ...)
```

Arguments

x	Numeric or formula (see examples). Vector with values of explanatory variable
y	Numeric. Vector with values of dependent variable. DEFAULT: NULL
data	Dataframe. If x is a formula, the according columns from data are used as x and y. DEFAULT: NULL
logy	Plot with a logarithmic y axis? Calls logAxis. DEFAULT: TRUE
predictnew	Vector with values to predict outcome for. Passed as newdata to predict.lm. DEFAULT: NULL
interval	Interval for prediction. DEFAULT: "confidence"
plot	Plot things at all? If FALSE, predictnew will still be returned. DEFAULT: TRUE
digits	Numeric vector of length ≥ 1. Specifies number of digits a,b,r,e are rounded to in the formula "y=a*log(x)+b, R^2, RMSE=e", respectively. If values are not specified, they are set equal to the first. DEFAULT: 2
inset	Numeric vector of length ≤ 2. inset distance(s) from the margins as a fraction of the plot region when formula is placed by keyword. DEFAULT: 0
xpd	Logical, specifying wheter formula can be written only inside the plot region (when FALSE) or inside the figure region including mar (when TRUE) or in the entire device region including oma (when NA). DEFAULT: par("xpd")
pos1	xy.coords-acceptable position of the formula. DEFAULT: "top"
pos2	For numerical coordinates, this is the y-position. DEFAULT: NULL, as in legend
add	Logical. If TRUE, line and text are added to the existing graphic. DEFAULT: FALSE (plots datapoints first and then the line.)
pch	Point Character, see par. DEFAULT: 16
col	Color of points, see par. DEFAULT: rgb(0,0,0, 0.5)
modcol	color of model line. DEFAULT: 2
lwd	Numeric. Linewidth, see par. DEFAULT: 1
xlab, ylab, main	Character / Expression. axis label and graph title if add=FAlSE. DEFAULT: internal from names
xlim, ylim	graphic range. DEFAULT: range(x)
...	Further arguments passed to plot and abline.

Value

predict.lm result.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec. 2014

See Also

lm, mReg, linReg.

Examples

```
x <- runif(100, 1, 10)
y <- 10^(0.3*x+rnorm(100, sd=0.3)+4)
plot(x,y)
expReg(x,y)
expReg(x,y, logy=FALSE)
expReg(x,y, predictnew=6, plot=FALSE)
expReg(x,y, predictnew=3:6, interval="none", plot=FALSE)
```

---

exTime *Time to run examples*

Description

Time the execution of examples. Useful in package development to identify functions taking much time.

Usage

```
exTime(topic = NA, package = NA, echo = FALSE, elapsed = FALSE,
  imagefile = TRUE, quiet = FALSE, ...)
```

Arguments

topic	Character string: the online help topic of which the examples should be run.
package	Charstring: installed and loaded package from which all examples should be run.
echo	Show the R input when sourcing? DEFAULT: FALSE
elapsed	Return *only* the third element (total elapsed time)? DEFAULT: FALSE
imagefile	Reroute graphics to pdf device? Will message the tempfile location if quiet=FALSE. DEFAULT: TRUE

quiet	Suppress warnings with both suppressWarnings and suppressMessages, also capture.output for str and cat results as well as setting pboptions(type="none") if pbapply is available.
...	Further arguments to example, especially run.dontrun, run.donttest and package, but NOT character.only and ask

Value

Time used as per system.time

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2016

See Also

example, system.time

Examples

```
exTime("plot")
exTime("yearSample", quiet=TRUE)
exTime(yearSample) # does NOT work, gives NULL and warning
exTime("yearSample", elapsed=TRUE, quiet=TRUE)
exTime("addFade", elapsed=TRUE, quiet=TRUE, run.dontrun=TRUE, run.donttest=TRUE)

## this takes quite some time if done for all functions in a package:
## Not run:
times <- exTime(package="rdwd")
as.matrix(sort(times))
system2("open", tempdir()) # to view the pdf graphics created by exTime

# times <- exTime(package="rdwd", run.dontrun=FALSE)

## End(Not run)
```

---

funnelPlot *Funnel plots for proportional data*

Description

Funnel plots for proportional data with confdence interval based on sample size. Introduced by Stephen Few, 2013

Usage

```
funnelPlot(x, n, labels = NULL, method = "classic", add = FALSE,
  xlim = range(n, finite = TRUE), ylim = range(x/n * 100, finite = TRUE),
  las = 1, xlab = "Sample size n", ylab = "Success rate [%]",
  main = "Funnel plot for Proportions", a3 = NULL, a2 = NULL, am = NULL,
  ap = NULL, at = NULL, al = NULL, ...)
```

Arguments

x	Numeric vector with number of successes (cases).
n	Numeric vector with number of trials (population).
labels	Labels for points. DEFAULT: NULL
method	Method to calculate Confidence interval, see "note" below. Can also be "wilson". DEFAULT: "classic"
add	Add to existing plot instead of drawing new plot? DEFAULT: FALSE
xlim	Graphical parameters, see par and plot. DEFAULT: range(n, finite=TRUE)
ylim	y limit in [0:1] DEFAULT: range(x/n*100, finite=TRUE)
las	DEFAULT: 1
xlab	DEFAULT: "Sample size n"
ylab	DEFAULT: "Success rate [%]"
main	DEFAULT: "Funnel plot for Proportions"
a3	List with arguments for CI lines at 3*sd (eg: col, lty, lwd, lend, etc.). Overwrites defaults that are defined within the function (if contentually possible). DEFAULT: NULL
a2	Arguments for line of 2 sd. DEFAULT: NULL
am	Arguments for mean line. DEFAULT: NULL
ap	Arguments for the data points (cex, etc.). DEFAULT: NULL
at	Arguments for text (labels of each point). DEFAULT: NULL
al	Arguments for legend (text.col, bty, border, y.intersp, etc.). DEFAULT: NULL
...	further arguments passed to plot only!

Value

Nothing - the function just plots

The basic idea

Salesman A (new to the job) has had 3 customers and sold 1 car. So his success rate is 0.33. Salesman B sold 1372 customers 632 cars, thus having a success rate of 0.46 Promoting B solely because of the higher rate fails to take experience and opportunity (n) into account! This dilemma is what the funnel plot with the confidence interval (ci) solves. See Stephen Few and Katherine Rowel's PDF for details on the interpretation.

Note

the default for lty is not taken from par("lty"). This would yield "solid". Overwriting lty for one of the three line categories then produces eg c("2", "solid", "solid"), which cannot be processed by legend.
Wilson's Method: algebraic approximation to the binomial distribution, very accurate, even for very small numbers.
http://www.apho.org.uk/resource/item.aspx?RID=39445 see "contains".
classic = Stephen Few's Method = the way I knew it: sqrt(mu*(1-mu) / n)
http://www.jerrydallal.com/LHSP/psd.htm
http://commons.wikimedia.org/wiki/File:ComparisonConfidenceIntervals.png
The apho Wilson method first yielded wrong upper limits in my translation (it needs 0:1 instead of %). Thus I added the wikipedia formula:
http://de.wikipedia.org/wiki/Konfidenzintervall_einer_unbekannten_Wahrscheinlichkeit#Wilson-Intervall
http://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval
Which other methods should I include? (That's not the hard part anymore)

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Oct 2013

References

http://www.perceptualedge.com/articles/visual_business_intelligence/variation_and_its_discontents.pdf
http://sfew.websitetoolbox.com/post/variation-and-its-discontents-6555336?
Excellent explanation of bayesian take on proportions: http://varianceexplained.org/r/empirical_bayes_baseball/

Examples

```
# Taken directly from Stephen Few's PDF:
funnel <- read.table(header=TRUE, text="
Name SampleSize Incidents
Tony 2 2
Mike 400 224
Jan 100 54
Bob 1000 505
Sheila 2 1
Jeff 10 5
Sandy 500 236
Mitch 200 92
Mary 10 3
John 2 0")

str(funnel)
X <- funnel$Incidents
N <- funnel$SampleSize

barplot(X/N, names=funnel$Name, main="success rate")
```

```
# not showing n!

funnelPlot(X,N)
# arguments for subfunctions as text may be given this way:
funnelPlot(x=X, n=N, labels=funnel$Name, at=list(cex=0.7, col="red"))
# Labeling many points is not very clear...
funnelPlot(X,N)
sel <- c(1,4,10) # selection
text(N[sel], (X/N*100)[sel], funnel$Name[sel], cex=0.7)
# You could also pass a vector with partly empty strings to funnelPlot
funnelPlot(x=X, n=N, labels=replace(funnel$Name, c(2,3,5:9), ""), at=list(adj=0.5))


# Even though Jan is more successfull than Mary in succes rate terms, both are
# easily within random variation. Mary may just have had a bad start.
# That Mike is doing better than average is not random, but (with 95% confidence)
# actually due to him being a very good seller.

# one more interesting option:
funnelPlot(X,N, a3=list(lty=2))

funnelPlot(X,N, a3=list(col=2, lwd=5))
# changing round line ends in legend _and_ plot is easiest with
par(lend=1)
funnelPlot(X,N, a3=list(col=2, lwd=5))

# The Wilson method yields slighty different (supposedly better) limits for small n:
funnelPlot(X,N, method="classic", al=list(title="Standard Method"))
funnelPlot(X,N, add=TRUE, method="wilson", a3=list(lty=2, col="red"),
           a2=list(lty=2, col="blue"), al=list(x="bottomright", title="Wilson Method"))

# Both Wilson method implementations yield the same result:
funnelPlot(X,N, method="wilson")
funnelPlot(X,N, add=TRUE, method="wilsonapho",
           a3=list(lty=2, col="red"), a2=list(lty=2, col="blue"))


# Note on nl used in the function, the n values for the ci lines:
plot(     seq(      10 ,       300 , len=50), rep(  1, 50) )
points(10^seq(log10(10), log10(300), len=50), rep(0.8, 50) )
abline(v=10)
# CI values change rapidly at small n, then later slowly.
# more x-resolution is needed in the first region, so it gets more of the points
```

---

funSource *Source code of a function*

Description

open source code of a function in a loaded or specified package on github.com/cran or github.com/wch/r-source

Usage

```
funSource(x, character.only = is.character(x), trydirect = TRUE)
```

Arguments

x	function name, with or without quotation marks
character.only	If TRUE, look for SomeFun instead of MyFun if MyFun <- "SomeFun". DEFAULT: is.character(x)
trydirect	If TRUE, try direct url to file x.R. DEFAULT: TRUE

Value

links that are also opened with browseURL

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan+Dec 2016

Examples

```
## Not run:  ## browser windows should not be openend in CRAN checks
library("berryFunctions")
funSource(colPoints)
funSource("head")
funSource("require", trydirect=FALSE)

funSource(earthDist)
funSource(OSMscale::earthDist)
funSource("OSMscale::earthDist")

## End(Not run)

## Not run:  # developmental testing
require(plotrix); require(scales)
funSource(rescale

tail <- function(...) stop("This is a dummy function. Type: rm(tail)")
funSource("tail")
rm(tail)

## End(Not run)
```

funTinn *Open function in TinnR*

Description

Opens function or object in external editor with an R command

Usage

```
funTinn(name)
```

Arguments

name	Name of function or object to be opened with the program associated with .r-files. In my case, the editor Tinn-R

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Aug 2014

See Also

edit, http://stackoverflow.com/questions/13873528

Examples

```
## Not run:
## Rcmd check --as-cran doesn't allow opening external devices,
## so this example is excluded from running in the checks.
## funTinn(boxplot.default)

## End(Not run)
```

getColumn *get column from data.frame*

Description

(Try to) extract a column from a data frame with USEFUL warnings/errors.
Watch out not to define objects with the same name as x if you are using getColumn in a function!

Usage

```
getColumn(x, df, trace = TRUE)
```

Arguments

x	Column name to be subsetted. The safest is to use character strings or substitute(input). If there is an object "x" in a function environment, its value will be used as name! (see upper2 example)
df	dataframe object
trace	Logical: Add function call stack to the message? DEFAULT: TRUE WARNING: in do.call settings with large objects, tracing may take a lot of computing time.

Value

Vector (or array, factor, etc) with values in the specified column

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sep 2016

See Also

subset, getElement, https://mran.revolutionanalytics.com/web/packages/car/vignettes/embedding.pdf

Examples

```
getColumn(Air.Flow, stackloss)
getColumn(2, stackloss)
getColumn("2",  stackloss) # works too...
getColumn(2,  stackloss[0,])
# The next examples all return errors:
try( getColumn(2,  stackloss[0])  )
try( getColumn(2,  stackloss[,0]) )
try( getColumn(Acid, stackloss)   ) # design choice: partial matching not supported
try( getColumn(2:3,  stackloss)   ) # cannot be a vector
try( getColumn(c("Air.Flow","Acid.Conc"),  stackloss)    )

upper <- function(x, select) getColumn(x, stackloss[select,])
upper(Water.Temp)
upper(2)
upper(2, select=0)
# upper(Water)  # error with useful message

# Pitfall lexical scoping: R only goes up until it finds things:
upper2 <- function(xx) {xx <- "Timmy!"; getColumn(xx, stackloss)} # will break!
is.error(      upper2(Water.Temp) ,   force=TRUE, tell=TRUE) # is an error

upper3 <- function(xx, dd) getColumn(substitute(xx), dd)
upper3(Air.Flow, stackloss) # may be safer in many scoping situations

# In packages use "colname" with quotation marks in level 2 functions to avoid
# the CRAN check NOTE "no visible binding for global variable"
```

```
df <- data.frame(x=letters[1:3],y=letters[4:6])
is.vector(df$x)
is.vector(getColumn("x", df)) # FALSE
# cannot force output to be a vector, as this will convert:
as.Date("2016-09-14")  ;  as.vector(as.Date("2016-09-14"))
# same problem with dfs from tapply results
# better ideas welcome!! (berry-b@gmx.de)

# Pitfall numerical column names:
df <- data.frame(1:5, 3:7)
colnames(df) <- c("a","1") # this is a bad idea anyways
getColumn("1", df) # will actually return the first column, not column "1"

getColumn(1, data.frame(AA=rep(NA,10)))
```

---

getName *get the name of an input in nested function calls*

Description

get the name of an input in nested function calls

Usage

```
getName(x)
```

Arguments

x	input object name or character string

Value

Character string with the name

Author(s)

http://stackoverflow.com/users/2725969/brodieg Implementation Berry Boessenkool, <berry-b@gmx.de>, Sep 2016

See Also

http://stackoverflow.com/a/26558733, substitute

Examples

```
# This does not work well:

lower <- function(x) deparse(substitute(x))
upper <- function(y) lower(y)
lower(pi) # returns "pi", as expected
upper(pi) # returns "y".

# That's why there is getName:

getName(pi) # returns "pi", as expected
upper <- function(y) getName(y)
upper(pi) # yay!

upper("dummy")
upper(dummy) # works also for nonexistent objects
dummy <- 7
upper("dummy") # still stable
upper(dummy) # still stable

upper(stackloss[1:5,])
```

---

gof *GOF measures*

Description

Goodness of Fit measures (GOF) for two vectors.
gofNA: not exported, checks input for each of the functions:
rsquare: Coefficient of determination (R2)
rmse: Root Mean Square Error (for minimising in optim)
nse: Nash-Sutcliffe efficiency, based on RHydro::eval.NSeff
kge: Kling-Gupta efficiency (better than NSE), based on hydroGOF::KGE, where there are many more options

Usage

```
gofNA(a, b, quiet = FALSE, fun = "")

rsquare(a, b, quiet = FALSE)

rmse(a, b, quiet = FALSE)

nse(a, b, quiet = FALSE)

kge(a, b, quiet = FALSE)
```

Arguments

a	Numerical vector with observational data
b	Simulated data (to be compared to a)
quiet	Should NA-removal warnings be suppressed? This may be helpful within functions. DEFAULT: FALSE
fun	Character string with function name for error and warning messages

Value

Single numerical value

Note

NAs are omitted with warning.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2016

See Also

cor, lm. http://en.wikipedia.org/wiki/R-squared, http://en.wikipedia.org/wiki/Mean_squared_error

Examples

```
# R squared and RMSE ------------------------------------------------------
set.seed(123)
x <- rnorm(20)
y <- 2*x + rnorm(20)
plot(x,y)
legGOF <- function(a,b)
  {
  text(a,b, paste(c("      R2","RMSE","  NSE","  KGE"), collapse="\n"), adj=1.2)
  text(a,b, paste(round(c(rsquare(x,y), rmse(x,y), nse(x,y), kge(x,y)),5),
                  collapse="\n"), adj=0)
  }
legGOF(-1.5, 2) # R2 good, but does not check for bias (distance from 1:1 line)

abline(a=0,b=1) ; textField(-1.5,-1.5, "1:1")
abline(lm(y~x), col="red")
p <- predict(lm(y~x))
points(x, p, pch=3, col="red")
segments(x, y, x, p, col="red")
stopifnot(all.equal(  nse(y,p) , rsquare(y,x)  ))


# Input checks
is.error(   rmse(1:6, 1:8)    ,  tell=TRUE)
nse(replace(x,3,NA), y)
```

```
kge(rep(NA,20), y)
rmse(0,0, quiet=TRUE)
rsquare(1:6, tapply(chickwts$weight, chickwts$feed, mean) )

# sample size bias
x <- 1:1000
y <- x+rnorm(1000)
rmse(x,y) # 0.983
ssize <- rep(5:1000, 3)
sgofs <- sapply(ssize, function(n){i <- sample(1:1000,n); c(rsquare(x[i],y[i]),rmse(x[i],y[i]))})
plot(ssize, sgofs[2,]) # RMSE: no bias, symmetric convergence
plot(ssize, sgofs[1,]) # R2: small underestimation in small samples


## Not run:  # time consuming Simulation
if(require(pbapply)) sapply <- pbsapply
r2 <- sapply(1:10000, function(i){
   x <- rnorm(20);  y <- 2*x + rnorm(20);  c(rsquare(x,y), rmse(x,y)) })
hist(r2[1,], breaks=70, col=5,
main= "10'000 times   x <- rnorm(20);  y <- 2*x + rnorm(20);  rsquare(x,y)")
# For small samples, R^2 can by chance be far off the 'real' value!
hist(r2[2,], breaks=70, col=5, main= "... rsquare(x,y)")
# RMSE is more symmetric and gaussian

## End(Not run)

# NSE and KGE ------------------------------------------------------------------

y <- dbeta(1:40/40, 3, 10) # simulated
x <- y + rnorm(40,0,sd=0.2) # observed
plot(x)
lines(y, col="blue")
legGOF(25, 2)
rmse(x,y) ; rmse(y,x)
nse(x,y) ; nse(y,x)  # x=obs, y=sim  (second command is wrong)
kge(x,y) ; kge(y,x)
```

`googleLink2pdf` *extract pdf link from google search result*

Description

restrict pdf link from a google search to actual link with text processing

Usage

```
googleLink2pdf(googlelink)
```

Arguments

googlelink Character string: A search result address

Value

Characterstring with only the basic link

Note

The function is not vectorized! If you have many links, use a loop around this function...

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012

See Also

strsplit, gsub

Examples

```
Link <- paste0("http://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1",
        "&cad=rja&sqi=2&ved=0CDIQFjAA&url=http%3A%2F%2Fcran.r-project.org",
        "%2Fdoc%2Fmanuals%2FR-intro.pdf&ei=Nyl4UfHeOIXCswa6pIC4CA",
        "&usg=AFQjCNGejDwPlor4togQZmQEQv72cK9z8A&bvm=bv.45580626,d.Yms")
googleLink2pdf(Link)

Link <- paste0("https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1",
        "&cad=rja&uact=8&ved=0ahUKEwjLlfmClavRAhWaN1AKHcGSBjEQFgghMAA",
        "&url=http%3A%2F%2Fstackoverflow.com%2Fquestions%2Ftagged%2Fr",
        "&usg=AFQjCNHYj6HjSs6Lvczn9wMWxE3slCdq1Q&bvm=bv.142059868,d.ZWM")
googleLink2pdf(Link)
Link <- paste0("https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2",
        "&cad=rja&uact=8&ved=0ahUKEwjLlfmClavRAhWaN1AKHcGSBjEQFggpMAE&",
        "url=http%3A%2F%2Fstackoverflow.com%2Fquestions%2Ftagged%2F%3Ftagnames",
      "%3Dr%26sort%3Dactive&usg=AFQjCNGkPGHq05qwKLLW4vRXdmk2Olhmig&bvm=bv.142059868,d.ZWM")
googleLink2pdf(Link)
```

---

groupHist *Histogramm for classes*

Description

Improvement of tapply(x, g, hist) with x and g taken from a data.frame

Usage

```
groupHist(df, x, g, xlab = "", ylab = "", las = 1, main = NULL,
  unit = NA, ...)
```

Arguments

df	data.frame object name
x	column name of variable of interest
g	column name of groups (INDEX in tapply, f in split)
xlab, ylab	axis labels. DEFAULT: ""
las	LabelAxisStyle, see par. DEFAULT: 1, means numbers on y-axis upright
main	Main title, internal default based on d, x, unit and g. DEFAULT: NULL
unit	Unit to be written into the default title. DEFAULT: NA
...	further arguments passed to hist

Details

Uses split to categorize into groups.

Value

NULL, used for plotting

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2015

See Also

hist, tapply

Examples

```
groupHist(chickwts, weight, "feed", col=2)
groupHist(chickwts, "weight", "feed", col=2, unit="grams at age 6 weeks")
groupHist(chickwts, weight, feed, col=2, breaks=20, main="Hi there")
groupHist(iris, Petal.Width, Species)
```

`headtail` *head and tail*

Description

show head and tail of an object with one command

Usage

```
headtail(x, n = 1, nh = n, nt = n, na = FALSE, ...)
```

Arguments

`x`	Object
`n`	Number of elements/rows/lines at begin and end of object to be returned. DEFAULT: 1
`nh, nt`	Number for `head` and `tail`, respectively. DEFAULT: n
`na`	Add NA values in between to emphasize visibly that there is something inbetween the values? DEFAULT: FALSE
`...`	Further arguments passed to `head` and `tail`

Details

Tries to find good methods of combining the two results acccording to codeclass(x).

Value

`head` result

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Mrz 2016

See Also

`head`

Examples

```
head(letters, n=3)
headtail(letters)
headtail(letters, n=3)
headtail(letters, n=3, na=TRUE)

head(letters, n=-10)
headtail(letters, n=-10, na=TRUE) # doesn't make sense for headtail
```

```
head(freeny.x, n=3)               # matrix
headtail(freeny.x, n=3, na=TRUE)  # no names for head-part
headtail(women, n=3, na=TRUE)     # data.frame works fine

head(freeny.y, n=3)
headtail(freeny.y, n=3, na=TRUE)

head(library, n=3)
headtail(library, n=3, na=TRUE)
headtail(library, na=TRUE)

ftable(Titanic)
head(stats::ftable(Titanic), n=4)
headtail(stats::ftable(Titanic), n=4, na=TRUE)

head(table(sample(1:9, 30, TRUE)), n=3)
headtail(table(sample(1:9, 30, TRUE)), n=3, na=TRUE)

head(table(state.division, state.region), n=3)
headtail(table(state.division, state.region), n=3, na=TRUE)
```

`horizHist` *Horizontal histogram*

Description

Draw a histogram with bars horizontally

Usage

```
horizHist(Data, breaks = "Sturges", freq = TRUE, plot = TRUE,
  col = par("bg"), border = par("fg"), las = 1, xlab = if (freq)
  "Frequency" else "Density", main = paste("Histogram of",
  deparse(substitute(Data))), ylim = range(HBreaks), labelat = pretty(ylim),
  labels = labelat, ...)
```

Arguments

`Data`	any data that `hist` would take.
`breaks`	character or numerical as explained in `hist`. DEFAULT: "Sturges"
`freq`	logical. if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). DEFAULT: TRUE
`plot`	logical. Should histogramm be plotted? FALSE to get just the hpos function. DEFAULT: TRUE
`col`	color. DEFAULT: par("bg")

border	color of borders of bars. DEFAULT: par("fg")
las	integer. Label axis style. DEFAULT: 1
xlab	character. Label for x-axis. DEFAULT: "absolute frequency"
main	character. Title for graphic. DEFAULT: "Histogram of substitute(Data)"
ylim	numerical vector of two elements. Y-axis limits. DEFAULT: range of data
labelat	numerical vector. Position of Y-Axis labels. DEFAULT: pretty(ylim)
labels	numerical or character. The labels themselves. DEFAULT: labelat
...	further arguments passed to barplot and axis

Details

Uses barplot to draw the histogram horizontally.

Value

function to address y-coordinates

Note

Doesn't work with breakpoints provided as a vector with different widths of the bars.
Please do not forget to use the function for vertical positioning from the **current** horizontal histogram. If It is not working correctly, you might have the function defined from some prior horizHist result.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011-2012

See Also

hist, barplot, axis

Examples

```
# Data and basic concept
set.seed(8); ExampleData <- rnorm(50,8,5)+5
hist(ExampleData)
hpos <- horizHist(ExampleData)
# Caution: the labels at the y-axis are not the real coordinates!
# abline(h=2) will draw above the second bar, not at the label value 2.
# Use hpos (horizontal position), the function returned by horizHist:
abline(h=hpos(11), col=2, lwd=2)

# Further arguments
horizHist(ExampleData, xlim=c(-8,20))
horizHist(ExampleData, ylab="the ... argument worked!", col.axis=3)
hist(ExampleData, xlim=c(-10,40)) # with xlim
horizHist(ExampleData, ylim=c(-10,40), border="red") # with ylim
```

```
hpos <- horizHist(ExampleData, breaks=20, col="orange")
axis(2, hpos(0:10), labels=FALSE, col=2) # another use of hpos()
```

`insertRows` *insert rows to data.frame*

Description

Insert (multiple) rows to a data.frame, possibly coming from another data.frame, with value and row recycling

Usage

```
insertRows(df, r, new = NA)
```

Arguments

`df`	data.frame
`r`	Row number (not name!), at which the `new` row is to be inserted. Can be a vector
`new`	Vector with data to be inserted, is recycled. Alternatively, a data.frame, whose rows are put into the r locations. If it has more rows than length(r), the excess rows are ignored. DEFAULT: NA

Value

data.frame

Note

Has not yet been tested with RWI (really weird input), so might not be absolutely foolproof

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Oct 2015, based on code by Ari B. Friedmann (I added the for loop, recycling, input controls and data.framification added by)

References

http://stackoverflow.com/questions/11561856/add-new-row-to-dataframe

See Also

addRows

Examples

```
existingDF <- as.data.frame(matrix(1:20, nrow=5, ncol=4))
existingDF
insertRows(existingDF, 2) # default new=NA is recycled
insertRows(existingDF, 2, 444:446)
insertRows(existingDF, 3, new=matrix(10:1,ncol=2)) # input warning
insertRows(existingDF, 1)
insertRows(existingDF, 5)
insertRows(existingDF, 6) # weird stuff...
insertRows(existingDF, 9) # not supposed to do that

# Works for multiple rows as well:
insertRows(existingDF, r=c(2,4,5), new=NA)
insertRows(existingDF, r=c(2,4,4), new=NA)

# Also works with a data.frame for insertion:
insertDF <- as.data.frame(matrix(101:112, nrow=3, ncol=4))
insertRows(existingDF, 3, new=insertDF) # excess rows in new are ignored
insertRows(existingDF, c(2,4,5), new=insertDF)
insertRows(existingDF, c(2,4:6), new=insertDF) # rows are recycled
```

`instGit` *install github package*

Description

Quickly install a package from github without having to install devtools with all its dependencies.

Usage

```
instGit(pk, cleanup = TRUE, ...)
```

Arguments

pk	Character string in the from of "user/package"
cleanup	Remove downloaded zipfile and folder with source code. DEFAULT: TRUE
...	Further arguments passed to `install.packages`, untested so far

Details

Works only for pure R package structure repositories from the master branch. Installs package dependencies listed in 'Imports' and 'Depends', but ignores version requirements! Tested only on windows 7 with R3.2.2. Note: `devtools::install_github` is much more extensive!
Note: drat is also much better than this quick hack. http://dirk.eddelbuettel.com/code/drat.html, https://github.com/eddelbuettel/drat, http://eddelbuettel.github.io/drat/DratForPackageAuthors.html Give your github users this code:

```
source("https://raw.githubusercontent.com/brry/berryFunctions/master/R/instGit.R")
instGit("brry/extremeStat")
library(extremeStat)
```

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2015 + Mar/Apr 2016

See Also

funSource, install_github in each of the packages devtools, ghit, remotes

Examples

```
if(FALSE){
instGit("talgalili/installr")
instGit("talgalili/installr", FALSE)
instGit("hadley/readxl")
instGit("mages/googleVis") # many dependencies!
instGit("twitter/AnomalyDetection")
instGit("yihui/knitr")
instGit("ramnathv/slidify")
instGit("jrnold/ggthemes")
}
```

---

is.error *Check if an expression returns an error*

Description

Does a given expression return an error? Useful for tests where you want to make sure your function throws an error.

Usage

```
is.error(expr, tell = FALSE, force = FALSE)
```

Arguments

expr	Expression to be tested for retunrning an error
tell	Logical: Should the error message be printed via message? DEFAULT: FALSE
force	Logical: Should an error be returned if the expression is not an error? DEFAULT: FALSE

Value

TRUE/FALSE

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2016

See Also

stop, try, inherits

Examples

```
is.error(  log(3)             )
is.error(  log("a")           )
is.error(  log(3),   tell=TRUE )
is.error(  log("a"), tell=TRUE )
stopifnot( is.error( log("a")  )  ) # or shorter:
is.error(  log("a"), force=TRUE)
# is.error(  log(3),   force=TRUE)
stopifnot(is.error(  is.error(log(3), force=TRUE)  ))
```

---

l2array *Convert list of arrays to array*

Description

Convert a list of arrays to a single array, conserving names. If dimnames do not need to be checked, you can also directly use
`do.call(abind::abind, list(LIST, rev.along=0, use.dnns=TRUE))`

Usage

```
l2array(x, ...)
```

Arguments

x	List with arrays. The dimension of the first is target dimension.
...	Further arguments passed to abind::abind

Value

array

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2016

See Also

l2df, help, http://stackoverflow.com/a/4310747

Examples

```
LIST <- lapply(LETTERS[1:5], function(x) array(paste0(x,1:24), dim=c(3,4,2)))
str(LIST)
LIST[[2]]
LISTa1 <- l2array(LIST)
LISTa1
str(LISTa1)

LISTm <- lapply(list(1:6,7:12,13:18,19:24), matrix, ncol=3,
               dimnames=list(x=c("a","b"), y=c("i","j","k"))  )
l2array(LISTm)

# The old l2array (<1.13.14, 2017-01-06) was very slow on large lists.
# I then found abind, which is much much much faster and easier on memory!
# It now replaces the internal old actual conversion code
# l2array still checks the dimnames
LISTa2 <- do.call(abind::abind, list(LIST, rev.along=0, use.dnns=TRUE))
LISTa2
stopifnot(all(LISTa1==LISTa2))
rm(LIST, LISTa1, LISTa2)


# General intro to arrays -----

A1 <- array(1:24, dim=c(4,2,3), dimnames=list(
                   my_x=paste0("row",1:4), my_y=c("A","B"), paste0("n",1:3)))
A1
which(A1==20, arr.ind=TRUE)

# Selection:
A1[,,"n2"]
A1[,,1:2]
A1["row2",,] # result rotated against expectation -> transpose with t(...)
A1[,"A",]
# aggregation:
apply(A1, MARGIN=1:2, FUN=sum) # keep first two dimensions
apply(A1, MARGIN=c(1,3), FUN=sum) # aggregate over my_y -> row1: 6, 22, 38
A1["row1",,]                                  # 1+5=6, 9+13=22, 17+21=38

as.vector(A1)

A <- array(1:24, dim=c(3,4,2), dimnames=list(x=paste0("x",1:3),
                                             y=paste0("y",1:4),
                                             z=paste0("z",1:2)))
str(A)
rm(A)
```

```
# l2array -----

A2 <- A1+2
A3 <- A1+4
LIST <- list(A1=A1, A2=A2, A3=A3) # list of arrays

LA <- l2array(LIST)
LA
str(LA)
LA[,,,"A2"]
LA["row2", ,"n2",]
avg <- apply(LA, MARGIN=1:3, mean)
stopifnot(all(avg==A2))


# names check -----

LISTN <- LIST
names(dimnames(LISTN[[2]]))[3] <- "intentional"
dimnames(LISTN[[3]])[3] <- list(paste0("k",1:3))
LAN <- l2array(LISTN)
LAN["row2", ,"k2",] # n2 is now changed to k2
LANa <- do.call(abind::abind, list(LISTN, rev.along=0, use.dnns=TRUE))
all(LAN==LANa)
str(LANa)

LISTN <- LIST
rownames(LISTN[[3]])[2] <- "intentional_diff"
LAN <- l2array(LISTN)

# data type check
is.error(   A <- l2array(c(LA, 999)),  tell=TRUE, force=TRUE)
```

`l2df` *List to data.frame*

Description

Convert list with vectors of unequal length to dataframe, pad with NAs

Usage

```
l2df(list, byrow = TRUE)
```

Arguments

`list`	List with vectors of irregular length.
`byrow`	Transposed output? DEFAULT: TRUE

Value

data.frame

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2014

References

http://stackoverflow.com/questions/5531471/combining-unequal-columns-in-r
http://stackoverflow.com/questions/15753091/convert-mixed-length-named-list-to-data-frame
http://stackoverflow.com/questions/5942760/most-efficient-list-to-data-frame-method
http://stackoverflow.com/questions/8799990/converting-given-list-into-dataframe
http://stackoverflow.com/questions/4227223/r-list-to-data-frame

See Also

l2array, sapply. If you have a LARGE list each with the same number of values, use the (much!) faster: plyr::quickdf.

Examples

```
eglist <- list(AA=c(6,9,2,6), BB=1:8, CC=c(-3,2) )
eglist
l2df(eglist)  # names are even kept
l2df(eglist, byrow=FALSE)
class(  l2df(eglist, byrow=FALSE)  ) # data.frame (since 2016-05-24)

eglist <- list(AA=c(6,9,2,6), BB="no", CC=c(-3,2) )
eglist
str(l2df(eglist))  # now everything is a character

eg2 <- list(AA=c(6,9,2,6), BB=matrix(1:8, ncol=2), CC=c(-3,2) )
eg2
l2df(eg2, FALSE)
# so a matrix is internally converted to a vector and then used regularly


# Naming ----

eg3 <- list(EE=c(AA=3.4),        FF=c(AA=3.5),        GG=c(AA=3.6))
eg4 <- list(EE=c(AA=3.4,BB=2.4), FF=c(AA=3.5,BB=2.5), GG=c(AA=3.6,BB=2.6))
l2df(eg3)
l2df(eg4)
l2df(eg3, byrow=FALSE)
l2df(eg4, byrow=FALSE)

eg3 <- list(c(AA=3.4),        c(AA=3.5),        c(AA=3.6))
eg4 <- list(c(AA=3.4,BB=2.4), c(AA=3.5,BB=2.5), c(AA=3.6,BB=2.6))
l2df(eg3)
```

```
l2df(eg4)
l2df(eg3, byrow=FALSE)
l2df(eg4, byrow=FALSE)

eg3 <- list(EE=c(3.4),     FF=c(3.5),     GG=c(3.6))
eg4 <- list(EE=c(3.4,2.4), FF=c(3.5,2.5), GG=c(3.6,2.6))
l2df(eg3)
l2df(eg4)
l2df(eg3, byrow=FALSE)
l2df(eg4, byrow=FALSE)

eg3 <- list(EE=c(3.4),     c(3.5),     c(3.6))
eg4 <- list(EE=c(3.4,2.4), c(3.5,2.5), c(3.6,2.6))
l2df(eg3)
l2df(eg4)
l2df(eg3, byrow=FALSE)
l2df(eg4, byrow=FALSE)


# Lists with dfs ----

eg5 <- list(AA=c(6,9,2,6), BB=data.frame(CC=1:8, DD=4:-3), EE=c(-3,2) )
eg5
is.error( l2df(eg5), tell=TRUE )# it is not possible to do this with a data.frame

# If you have a list with only data.frames, you could use the following:
eg6 <- list(AA=data.frame(BB=1:8, CC=4:-3), DD=data.frame(EE=23:24, FF=c(-3,2)))
eg6
do.call(cbind, eg6) # but this recycles the values of shorter tables!
names(eg6$DD) <- names(eg6$AA)
do.call(rbind, eg6)
# check some of the links above for more solutions...
```

`legendmt` *legend with multiline title*

Description

Draw a legend with title spanning several lines (i.e. with line breaks). Note that this is in development and not all inputs ar correctly vectorized yet.

Usage

```
legendmt(x, y = NULL, legend, title, x.intersp = 1, fill = NA,
  col = par("col"), border = NA, lty = NA, lwd = NA, pch = NA, ...)
```

Arguments

x, y, legend	Arguments as in legend
title	Character with linebreaks or vector of charstrings.
x.intersp, fill, col, border, lty, lwd, pch	Arguments as in legend
...	Further arguments passed to legend. If vectorized, please remember to prepend NAs or whatever.

Value

legend output

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Apr 2017

See Also

legend

Examples

```
plot(1:10)
  legend("topleft", letters[1:4], col=1:4, pch=1, title="very long title to be split up")
legendmt("topleft", letters[1:4], col=1:4, pch=1, title="very long title\nnow splat up")

# Alternative:
plot(1:10)
legend("topleft", "very long title to be split up")
legend("topleft", letters[1:4], col=1:4, pch=1, inset=c(0,0.09) )
```

---

library2 *install.package and require*

Description

install and load a package. If a package is not available, it is installed before being loaded

Usage

```
library2(name, libargs = NULL, ...)
```

Arguments

name	Name of the package(s). Can be qouted, must not.
libargs	List of arguments passed to library like lib.loc, quietly etc. DEFAULT: NULL
...	Arguments passed to install.packages like lib, repos etc.

Value

messages help instruction.

Note

Passing a vector with packages will work, but give some warnings.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2014

See Also

install.packages, library

Examples

```
## Not run:
## Excluded fom CRAN checks. Package installation on server is unnecessary.
require2(ada)
library2("statmod")

## End(Not run)
```

---

lim0 *axis limits with one end at zero*

Description

Calculates the range needed for ylim or xlim in plot, so that axis starts at zero and is extended by 4% at the other end

Usage

```
lim0(x, f = 1/27, curtail = TRUE)
```

Arguments

x	Numeric. Vector with values
f	Numeric. Extension factor. DEFAULT: 0.04 as in extendrange used eg. by curve
curtail	Logical. Should the range returned be trimmed by 4%? That way, plotting doesn't need the default par xaxs or yaxs changed. DEFAULT: TRUE

Value

Vector with two values: 0 and by 4

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 6.6.2013

References

methods(plot), plot.default. Actually, I found extendrange via plot.function in curve

See Also

The extendrange() utility in package **grDevices**

Examples

```
# basic idea:
val <- c(3.2, 1.8, 4.5, 2.8, 0.1, 2.9) # just some numbers
plot(val, ylim=lim0(val) ) # you don't even have to set yaxs="i" ;-)

# "normal" plot:
plot(val)
par("usr")  # -0.076  4.676

# if y-axis is not allowed to go below 0, and we're too lazy to set yaxs="i":
plot(val, ylim=lim0(val) )
round( par("usr")  , digits=5) # 0.00000 4.66296

# with 0.04 extension as claimed by help page (1/27 in source code = 0.037):
plot(val, ylim=lim0(val, f=0.04) )
round( par("usr")  , digits=5) # zero is not included on axis anymore

b <- -val
plot(b)
plot(b, ylim=lim0(b) ) # works with only negative values as well
```

linLogHist *lin-log transition histogram*

Description

Draw histograms that gradually transform from a linear to a logarithmic axis (animation)

Usage

```
linLogHist(x, steps = 100, breaks = 20, col = "blue", las = 1,
  xlab = deparse(substitute(x)), xlim = range(x, finite = TRUE),
  box = TRUE, parexpr, endexpr, sleep = 0, axisargs = NULL,
  axisargs2 = NULL, firstplot = TRUE, lastplot = TRUE, write_t = TRUE,
  values_t = NULL, ...)
```

Arguments

x	x values to be plotted in animation
steps	Number of steps in transition. DEFAULT: 100
breaks	hist breaks. DEFAULT: 20
col	hist color. DEFAULT: "blue"
las	par LabelAxisStyle (numbers upright). DEFAULT: 1
xlab	Label for the x axis. DEFAULT: deparse(substitute(x))
xlim	xlim range in non-log units. DEFAULT: range(x, finite=TRUE)
box	Draw box at the end to overplot ablines crossing the box? DEFAULT: TRUE
parexpr	Characterized Expression to set par, eg. parexpr='par(mar=c(2,0.5,1.5,0.5), mpg=c(1.8,1,0))'
endexpr	Characterized Expression executed at the end of the plot, eg. endexpr='mtext("Probability Density"
sleep	Pause time between frames, in seconds, passed to Sys.sleep. DEFAULT: 0
axisargs	List of arguments passed to logVals, like base. DEFAULT: NULL
axisargs2	List of arguments passed to logAxis in the final plot. DEFAULT: NULL
firstplot	plot on linear scale first? DEFAULT: TRUE
lastplot	plot on logarithmic scale at the end? DEFAULT: TRUE
write_t	write transformation value in lower right corner? DEFAULT: TRUE
values_t	Supply vector with values for transformation (1/t). Overides steps. If you have a better algorithm than I do, please let me know! DEFAULT: NULL
...	further arguments passed to hist, like freq, main, xlim, ylab. Excluded: x, xaxt, possibly add

Value

Returned invisibly: transformation values used. Plotted: steps number of images.

Note

It's best to save the plots into a pdf or wrap it within
`png("Transition%03d"); linLogHist(x); dev.off()`

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, April 2015

See Also

linLogTrans

Examples

```
x <- rlnorm(700, m=3)
hist(x, col=4)
hist(log10(x), xaxt="n"); logAxis(1); hist(log10(x), col=4, add=TRUE)

op <- par()
linLogHist(x, steps=8, sleep=0.01) # 0.05 might be smoother

linLogHist(x, xlab="ddd", breaks=30, steps=3, write_t=FALSE, yaxt="n", freq=FALSE,
   main="", parexpr='par(mar=c(2,0.5,1.5,0.5), mgp=c(1.8,1,0))',
   endexpr='mtext("Probability Density", line=-1.2, adj=0.03, outer=T)')
par(op)

## Not run:
## Rcmd check --as-cran doesn't like to open external devices such as pdf,
## so this example is excluded from running in the checks.
pdf("LinLogTransitionAnimation.pdf")
linLogHist(x, main="Example Transition", steps=20, freq=FALSE)
dev.off()

# if you have FFmpeg installed, you can use the animation package like this:
library2(animation)
saveVideo(linLogHist(x, steps=50), video.name="linlog_anim.mp4", interval=0.08,
ffmpeg="C:/ffmpeg-20150424-git-cd69c0e-win64-static/bin/ffmpeg.exe")

## End(Not run)
```

---

linLogTrans *Animation for transition from linear to logarithmic axis*

Description

draw images that gradually transform from a linear to a logarithmic axis

Usage

```
linLogTrans(x, y, log = "x", steps = 100, base = 1, las = 1,
  plot = TRUE, xlim = range(x, finite = TRUE), ylim = range(y, finite =
  TRUE), box = TRUE, parexpr, endexpr, sleep = 0, firstplot = TRUE,
  lastplot = TRUE, write_t = TRUE, values_t = NULL, pointsarg = NULL,
  ...)
```

Arguments

x	x values to be plotted in animation
y	Vector with corresponding y values
log	Which axis is logarithmic, "x" or "y". DEFAULT: "x"
steps	Number of steps (images) in transition (About 30% are taken out). DEFAULT: 100
base	Base passed to logVals. DEFAULT: 1
las	par LabelAxisStyle (numbers upright). DEFAULT: 1
plot	Plot animations at all? False to just get the t-vector (used in linLogHist). DEFAULT: TRUE
xlim	xlim range in non-log units. DEFAULT: range(x, finite=TRUE)
ylim	ylim range in non-log units. DEFAULT: range(y, finite=TRUE)
box	Draw box at the end to overplot ablines crossing the box? DEFAULT: TRUE
parexpr	Characterized Expression to set par, eg. parexpr='par(mar=c(2,0.5,1.5,0.5), mpg=c(1.8,1,0))'
endexpr	Characterized Expression executed at the end of the plot, eg. endexpr='mtext("Probability density"
sleep	Pause time between frames, in seconds, passed to Sys.sleep. DEFAULT: 0
firstplot	Plot data on linear axis as additional first image? DEFAULT: TRUE
lastplot	Plot data on logarithmic axis as additional last image? DEFAULT: TRUE
write_t	Write transformation value in lower right corner? DEFAULT: TRUE
values_t	Supply vector with values for transformation (1/t). Overides steps. If you have a better algorithm than I do, please let me know! DEFAULT: NULL for internal calculation based on size of steps.
pointsarg	List of further arguments passed to points, like pch, cex, col. DEFAULT: NULL
...	Further arguments passed only to plot, like main, xlim, ylab. Excluded: x, y, las, xaxt, type

Value

Returned invisibly: transformation values used. Plotted: steps number of images.

Note

if(steps>1000) steps <- 1000. In the unlikely case you need more steps, please let me know and I'll change the code.
It's best to save the plots into a pdf (see the example) or wrap it within
png("Transition%03d"); linLogTrans(x,y); dev.off()

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, June 2014

References

x^(1/t) is based on the first comment on http://stackoverflow.com/questions/15994442/
besides the nice graphic properties of logtransformations, check this page for the implications on rates of change:
http://sfew.websitetoolbox.com/post/show_single_post?pid=1282690259&postcount=4
http://sfew.websitetoolbox.com/post/show_single_post?pid=1282691799&postcount=5

See Also

logVals

Examples

```
set.seed(42);  x <- 10^rnorm(100, 3);  y <- runif(100)
linLogTrans(x,y, steps=15, sleep=0.01) # 0.05 might be smoother...
linLogTrans(x,y, steps=15, log="y", ylim=c(0.1, 0.8), base=c(1,2,5))

## Not run:
## Rcmd check --as-cran doesn't like to open external devices such as pdf,
## so this example is excluded from running in the checks.
pdf("LinLogTransitionAnimation.pdf")
linLogTrans(x,y, main="Example Transition")
dev.off()

# if you have FFmpeg installed, you can use the animation package like this:
library2(animation)
saveVideo(linLogTrans(x,y, steps=300), video.name="linlog_anim.mp4", interval=0.01,
    ffmpeg="C:/ffmpeg-20150424-git-cd69c0e-win64-static/bin/ffmpeg.exe")


# old t values were dependent on the value of steps
findt <- function(steps) {
  # t-values for x^(1/t):
  allt <- 10^(seq(0,2.5,len=1e4) )
  # selection at upper half of these values;
  # Otherwise, the animation slows down too much at the end
  f <- 1.4 # multiplication factor due to length loss by unique
  sel <- round(seq(1, 10, len=f*steps)^4)   #0.5*seq(1, 100, len=1.3*steps)^2 + 0.5*
  sel2 <- unique(round(log10(seq(1, 10, len=f*steps))*f*steps))
  sel2[1] <- 1
  sel <- sel[sel2]
  # final t-values for transition:
  allt <- unique(round(allt[sel], 2))
  data.frame(x=seq(1,1000,len=length(allt)), t=allt)
  }
```

```
plot(findt(1000), type="l", log="y", las=1)
for(i in 5:999) lines(findt(i), col=rainbow2(1000)[i])
d <- findt(300)
lines(d) # good average

plot(d$x[-1], diff(d$t), type="l", ylim=c(3e-3,10), yaxt="n", log="y", main="t value growth rate")
logAxis(2) ; lines(d$x[-1], diff(d$t))
d2 <- findt(1000)
lines(d2$x[-1], diff(d2$t), col=2)
lines(2:1000, diff(linLogTrans(1,1, steps=1000, plot=F)), col=4)


d <- findt(300)
pdf("degreepoly.pdf")
for(i in 5:30)
  {
  plot(d, log="y", type="l", lwd=3, main=i, xlim=c(0,300), ylim=c(1,2))
  modell <- lm(t ~  poly(x,i, raw=T), data=d)
  lines(x2, predict(modell, data.frame(x=1:1300)), col=2)
  }
dev.off()   # 17 is good

cf <- coef(lm(t ~  poly(x,17, raw=T), data=d)) # these are currently used in the function
x <- 1:1000
y <- rowSums(sapply(1:18, function(i) cf[i]*x^(i-1)), na.rm=TRUE)
lines(x, y, lwd=3)
y[1] <- 1
plot(x, round(y, 3), ylim=c(1,3), xlim=c(0,500), type="l", log="")
dput(round(y, 3))

findn <- function(steps) nrow(findt(steps))
plot(1:1000, sapply(1:1000, findn), type="l")
abline(b=1, a=0)


## End(Not run)
```

---

linReg *linear regression with plotting*

Description

uses lm; plots data if add=FALSE, draws the regression line with abline and writes the formula with legend

Usage

```
linReg(x, y = NULL, data = NULL, add = FALSE, digits = 2, pch = 16,
  col = 2, colband = addAlpha(col), level = 0.95, lwd = 1,
```

```
xlab = deparse(substitute(x)), ylab = deparse(substitute(y)),
main = "linear regression", pos1 = "top", pos2 = NULL, inset = 0,
legargs = NULL, ...)
```

Arguments

x	Numeric or formula (see examples). Vector with values of explanatory variable
y	Numeric. Vector with values of dependent variable. DEFAULT: NULL
data	Dataframe. If x is a formula, the according columns from data are used as x and y. DEFAULT: NULL
add	Logical. If TRUE, line and text are added to the existing graphic. DEFAULT: FALSE (plots datapoints first and then the line.)
digits	Numeric vector of length ≥ 1. Specifies number of digits a,b,r,e are rounded to in the formula "y=a*x+b \n R^2=r \n RMSE=e", respectively. If values are not specified, they are set equal to the first. DEFAULT: 2
pch	Point Character of datapoints, see par. DEFAULT: 16
col	Color of the regression line, see par. DEFAULT: 2
colband	Color of the confidence region band. DEFAULT: addAlpha(col)
level	Confidence level, see predict.lm. DEFAULT: 0.95
lwd	Numeric. Linewidth, see par. DEFAULT: 1
xlab	Axis label if add=FALSE. DEFAULT: deparse(substitute(x))
ylab	Axis label if add=FALSE. DEFAULT: deparse(substitute(y))
main	Title if add=FALSE. Changed (if not specified) for x=formula with data. DEFAULT: "linear regression"
pos1	xy.coords-acceptable position of the formula. DEFAULT: "top"
pos2	For numerical coordinates, this is the y-position. DEFAULT: NULL, as in legend
inset	Numeric vector of length ≤ 2. inset distance(s) from the margins as a fraction of the plot region when formula legend is placed by keyword. DEFAULT: 0
legargs	list of arguments passed to legend, like list(cex=0.8, xpd=TRUE, bg="white"), ... xpd specifies whether formula can be written only inside the plot region (when FALSE) or inside the figure region including mar (when TRUE) or in the entire device region including oma (when NA). DEFAULT: NULL
...	Further arguments passed to plot and abline.

Value

None, used for plotting and drawing.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011-2012, 2015

See Also

lm, mReg, expReg, legend, par, abline.

Examples

```
a <- 1:30
b <- a/2.345+rnorm(30,0,3)

linReg(a,b)
linReg(a,b, ylab="Hallo", pch=1, col=3, main="Regression by Berry")
linReg(a, b, pos1=15, pos2=0) # position of topleft corner of legend
linReg(a, b, pos1=NA, col="orange") # to suppress legend

# Formula specification:
linReg(b~a)
linReg(Volume~Height, data=trees)

# For more flexibility with the datapoints, plot first, then use linReg with add=TRUE:
plot(a,b, xlim=c(-5,45))
linReg(a, b, pos1="bottomright", add=TRUE, inset=.1) # inset: distance from plot border
linReg(a, b, digits=c(7,4,3), add=TRUE, col=3, lty=2, lwd=4, level=0.8)
linReg(a, b, pos1="topleft", inset=c(-0.1, 0.3), legargs=list(xpd=TRUE), add=TRUE)
```

---

locArrow *arrow at locator point in graph*

Description

Draw arrow at positions in a graph located by clicking and return the code to recreate it

Usage

```
locArrow(digits = 2, length = 0.1, code = 2, ...)
```

Arguments

digits	Number of digits coordinates are rounded to with signif
length	Length of the edges of the arrow head (in inches). DEFAULT: 0.1
code	Direction of arrow head. DEFAULT: 2 (from first to last point clicked)
...	Further arguments passed to arrows like lwd, col etc

Details

Not tested across platforms yet...

Value

Character string with code

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jun 2016

See Also

locLine, locator, abline

Examples

```
plot(cumsum(rnorm(60)), type="l")
## locArrow() # only do this manually in interactive() mode
## locArrow(col="blue", lwd=3)
```

---

locLine *abline at locator point in graph*

Description

Draw vertical and/or horizontal lines at positions in a graph located by clicking

Usage

```
locLine(h = TRUE, v = TRUE, n = 1, ...)
```

Arguments

h	Draw horizontal line at clicked location? DEFAULT: TRUE
v	Draw vertical line at clicked location? DEFAULT: TRUE
n	Number of points to be clicked. DEFAULT: 1
...	Further arguments passed to abline like lty, lwd, col, etc

Details

Not tested across platforms yet...

Value

locator result

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Mar 2016

See Also

locator, abline

Examples

```
plot(cumsum(rnorm(60)), type="l")
## locLine() # only do this manually in interactive() mode
```

---

logAxis *Label logarithmic axes*

Description

Shortcut to calling logVals, axis and abline

Usage

```
logAxis(side = 1, log = NULL, lcol = "grey", lty = 1, lwd = 1,
  labels = NULL, allticks = FALSE, allargs = NULL, expr, las = 1, from,
  to, Range, base, big.mark = "'", decimal.mark = ".", scientific = FALSE,
  exponent = 5, expobase1 = FALSE, allbase = 1:9, box = TRUE, ...)
```

Arguments

side	Which axis are to be labeled? Can be a vector within 1:4. DEFAULT: 1
log	Is the axis logarithmic by plot(log="x")? internal DEFAULT: par("xlog") or "ylog". DEFAULT: NULL
lcol	Color of gridlines drawn in the graph with abline, NA to suppress. DEFAULT: "grey"
lty, lwd	Type of gridlines. DEFAULT: 1
labels	Labels passed to axis. "FALSE" to suppress labelling. DEFAULT: NULL (internally, logVals$labs
allticks	Place all intermediate ticklines at the axis (without labelling). DEFAULT: FALSE
allargs	List of arguments passed to axis for allticks=TRUE. DEFAULT: NULL
expr	Expression drawing over the ablines, like (points(x,y). Can be code within braces.
las	LabelAxisStyle for the orientation of the labels. DEFAULT: 1
from	Lower exponent OR vector with data, as in logVals. DEFAULT based on par("usr")
to	High end exponent. DEFAULT: internally based on par("usr")
Range	Override from and to as range.
base	Bases to be used in logVals. DEFAULT: c(1,2,5) or 1, depending on from and to.

big.mark	Symbol separating thousands, eg. space, comma, dot, etc. see "format" and "prettyNum". DEFAULT: "'"
decimal.mark	Character separating comma values, see "format" and "prettyNum". DEFAULT: "."
scientific	See format. DEFAULT: FALSE
exponent	Starting at which exponent should logVals return an expression with exponents? DEFAULT: 5
expobase1	Should "n * " be appended before 10^exp if n=1? DEFAULT: FALSE
allbase	base for $all (for horizontal lines). DEFAULT: 1:9
box	Draw box at the end to overplot ablines crossing the box? DEFAULT: TRUE
...	Further arguments passed to axis, like lwd, col.ticks, hadj, lty, ...

Value

An invisible list with

vals	Values for lines and label positions
labs	Formatted values for labels
all	Values for lines

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

See Also

logVals, log10

Examples

```
x <- 10^runif(200, -1, 2)
plot(x, yaxt="n", log="y", pch=16)
logAxis(2)
# overplot vertical lines:
logAxis(2, expr=points(x, pch=16), base=1, col.axis=4, font=2)

# plots where log="x" is not possible:
hist(log10(x), breaks=20, col.axis="grey", main="")
logAxis(side=3, expr=hist(log10(x), breaks=20, add=TRUE, col=3))
# or just use the new logHist function (Feb 2016):
logHist(x, breaks=20, col=3)

# automatic calculation of from, to and base:
plot(1:3, axes=FALSE)
logAxis(1:2) # side can be a vector - nice, huh?
plot(-1:4, axes=FALSE)
logAxis(1:2) # treshold for base 1 instead of c(1,2,5) at 4 exponents exceeded.
```

```
plot(1:3, axes=FALSE)
logAxis(1:2, allticks=TRUE, lcol=NA)

par(mar=c(3,3,1,4))
plot(8:15) ;  logAxis(4) # with exponents if they are above 5
plot(10^(1:4), ylim=10^c(4,1), type="o", log="y") # reverse axis:
plot(10^(1:5), log="y"); logAxis(4, exponent=3) # different treshold
plot(10^(1:5), log="y"); logAxis(4, expon=3, base=c(1,2,5), expobase1=TRUE)
plot(-8:5); logAxis(4, allbase=c(1,2,5)) # In case you want to mislead...
```

`logHist` *Histogram of logarithmic values*

Description

Draw histogram of values on a logarithmic scale with nice axis labels

Usage

```
logHist(x, logargs = NULL, main = xmain, xlab = xname, col = "tan",
  add = FALSE, las = 1, ylim = NULL, freq = TRUE, quiet = FALSE, ...)
```

Arguments

`x`	Vector of numerical values
`logargs`	A list of arguments passed to `logAxis`. DEFAULT: NULL
`main`	Title of graph, internally from x. DEFAULT: internal name representation
`xlab`	X axis label. DEFAULT: internal: name of x
`col`	Color of histogram bars
`add`	Logical: add to existing plot?
`las`	Integer: label axis style. DEFAULT: 1 (numbers upright)
`ylim`	2 Numbers: y-axis range. DEFAULT: NULL
`freq`	Logical: counts instead of density? DEFAULT: TRUE
`quiet`	Logical: suppress warning about non-positive values? DEFAULT: FALSE
`...`	further arguments passed to `hist` like breaks, xlim=c(-1,3), ..., but not xaxt

Value

none

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2016

See Also

logAxis, hist

Examples

```
dat <- rbeta(1e4, 2, 18)*100
hist(dat, col="tan", breaks=50)
logHist(dat)
logHist(dat, freq=FALSE)
logHist(dat, breaks=50)
logHist(dat,xlim=c(0,2)) # xlim in powers of ten
logHist(c(-1,0,1,2,2,3,3,4,8,10,50)) # warning for negative values
```

---

logSpaced *Logarithmically spaced points*

Description

Calculates values that are in logarithmic distance from each other e.g. to produce logarithmic interval borders

Usage

```
logSpaced(base = 1.1708, n = 20, min = 1, max = n, plot = TRUE,
  pch = 3, las = 1, ylab = "base", ...)
```

Arguments

base	Base for calculations, can be a vector to compare several bases. DEFAULT: 1.1708
n	Number of values to be calculated. DEFAULT: 30
min, max	Range where n values are to be distributed, single values each. DEFAULT: 1,n
plot	Should the points be plotted on a line? DEFAULT: TRUE
pch, las	PointCharacter and Label Axis Style. DEFAULT: 3,1
ylab	Y axis label. DEFAULT: "base"
...	Further arguments passed to plot

Value

Vector or matrix, depending on base input

Note

base >1 concentrates points at low values, base<1 at high values. base does not relate to base in log!

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Oct 2014

See Also

classify, log

Examples

```
logSpaced()
logSpaced(base=c(1.1, 1.5, 2), n=6, min=5, max=10)
d <- logSpaced(seq(0.8, 1.2, 0.025), main="logarithmically spaced points")

# the default base for the default n (20) will give an approximately equal
# bin width across the range on a logarithmic scale:
d <- logSpaced()
plot(d, rep(1,20), log="x")
```

---

logVals *Create log-axis values and labels*

Description

Create nice values and labels to write at logartihmic axes

Usage

```
logVals(from = -7, to = 7, Range, base = 1, big.mark = "'",
  decimal.mark = ".", scientific = FALSE, exponent = Inf,
  expobase1 = FALSE, allbase = 1:9, ...)
```

Arguments

from	Lower exponent *OR* vector with data
to	High end
Range	Or give from and to as range
base	Bases to be used, eg. c(1,2,5)
big.mark	Symbol separating thousands, eg. space, comma, dot, etc. see format and prettyNum
decimal.mark	Character separating comma values, see format and prettyNum
scientific	See format
exponent	Starting at which exponent should labs be an expression with exponents? Compare to options("scipen"). This is mainly for logAxis and only for base 1. DEFAULT: Inf

expobase1	Should "n * " be appended before 10^exp if n=1? DEFAULT: FALSE
allbase	Base for $all (for horizontal lines). DEFAULT: 1:9
...	Ignored arguments

Value

A list with

vals	Values for lines and label positions
labs	Formatted values for labels
all	Values for lines

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2014

See Also

log10, logAxis, http://r.789695.n4.nabble.com/expression-exponent-labeling-td4661174.html

Examples

```
# Easiest use: vector with data (logVals automatically finds range):
y <- 10^runif(50, -1, 2)
plot(y, log="y") # not much control over placement and format of labels
plot(y, log="y", yaxt="n")
# now do this better, with custom bases:
lv <- logVals(y, base=c(1,2,5) )
axis(2, lv$vals, lv$labs, las=1)

# Default arguments:
lv <- logVals()
str(lv) # values, formatted labels, all 10^x values for lines
plot(1, ylim=c(1e-3, 1e4), log="y", yaxt="n", yaxs="i")
abline(h=lv$all, col=8 )
box("plot")
axis(2, lv$vals, lv$labs, las=1)
lines(seq(0.5, 1.5, len=50), 10^runif(50, -3, 4), col=2)

# Formatting labels:
logVals(                )$labs
logVals(scient=TRUE     )$labs
logVals(exponent=5      )$labs # expression with exponent, see logAxis
logVals(big.mark=" "    )$labs
logVals(big=".", dec=",")$labs # German style (not recommended)
```

lsc *Linear storage cascade, unit hydrograph*

Description

Optimize the parameters for unit hydrograph as in the framework of the linear storage cascade. Plot observed & simulated data

Usage

```
lsc(P, Q, area = 50, Qbase = Q[1], n = 2, k = 3, x = 1:length(P),
  fit = 1:length(Q), plot = TRUE, main = "Precipitation and discharge",
  plotsim = TRUE, returnsim = FALSE, type = c("o", "l"),
  legx = "center", legy = NULL, ...)
```

Arguments

P	Vector with precipitation values **in mm in hourly spacing**
Q	Vector with observed discharge (runoff) **in m^3/s** with the same length as precipitation.
area	Single numeric. Catchment area **in km^2**
Qbase	baseflow that is added to UH-induced simulated Q, thus cutting off baseflow in a very simple manner.
n	Numeric. Initial number of storages in cascade. not necessarily integer. DEFAULT: 2
k	Numeric. Initial storage coefficient (resistance to let water run out). High damping, slowly reacting landscape, high k. DEFAULT: 3
x	Vector for the x-axis of the plot. DEFAULT: sequence along P
fit	Integer vector. Indices for a subset of Q that Qsim is fitted to. DEFAULT: all of Q
plot	Logical. plot input data? DEFAULT: TRUE
main	Character string. DEFAULT: "Precipitation and discharge"
plotsim	Logical. add best fit to plot? DEFAULT: TRUE
returnsim	Logical. Return simulated Q instead of parameters of UH? DEFAULT: FALSE
type	Vector with two characters: type as in plot, repeated if only one is given. 1st for obs, 2nd for sim. DEFAULT: c("o","l")
legx	legend position. DEFAULT: "center"
legy	legend position. DEFAULT: NULL
...	arguments passed to optim

Value

Either vector with optimized n and k and the Nash-Sutcliffe Index, *or* simulated discharge, depending on the value of returnsim

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2013

References

http://ponce.sdsu.edu/onlineuhcascade.php
Skript 'Abflusskonzentration' zur Vorlesungsreihe Abwasserentsorgung I von Prof. Krebs an der TU Dresden
http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_forst_geo_und_hydrowissenschaften/fachrichtung_wasserwesen/isiw/sww/lehre/dateien/abwasserbehandlung/uebung_ws09_10/uebung_awe_1_abflusskonzentration.pdf
http://www.uni-potsdam.de/fs-g3/file.php?fileserver=klausuren&file=%2FMaster_of_Science%2FHydroII_Lernzettel.pdf

See Also

unitHydrograph, superPos, nse, rmse. deconvolution.uh in the package hydromad, http://hydromad.catchment.org

Examples

```
qpfile <- system.file("extdata/Q_P.txt", package="berryFunctions")
qp <- read.table(qpfile, sep="\t", dec=",", header=TRUE)
calib <- qp[1:90,]
valid <- qp[-(1:90),]

# Area can be estimated from runoff coefficient (proportion of N becoming Q):
#    k*P * A = Q * t      A = Qt / kP
# Q=0.25 m^3/s  * t=89 h   *  3600 s/h   k=psi* P =34mm = 0.034m = m^3/m^2
#                                                    / 1e6 m^2/km^2   = km^2
mean(calib$Q) * length(calib$Q) *3600  / ( 0.7 * sum(calib$P)/1000) / 1e6
# 3.368 km^2


# calibrate Unit Hydrograph:
UHcalib <- lsc(calib$P, calib$Q, area=3.4)
UHcalib # n 0.41  k 244.9  NSE 0.74  psi 0.45
# Psi is lower than 0.7, as it is now calculated on direct runoff only

# Corresponding Unit Hydrograph:
UH <- unitHydrograph(n=UHcalib["n"], k=UHcalib["k"], t=1:length(calib$P))
plot(UH, type="l") # That's weird anyways...
sum(UH) # 0.58 - we need to look at a longer time frame

# calibrate Unit Hydrograph on peak only:
lsc(calib$P, calib$Q, area=3.4, fit=17:40) # n 0.63  k  95.7  NSE 0.67
# for fit, use index numbers, not x-axis units (if you have specified x)

# Simulated discharge instead of parameters:
lsc(calib$P, calib$Q, area=3.4, returnsim=TRUE, plot=FALSE)
```

```
# Apply this to the validation event
dummy <- lsc(valid$P, valid$Q, area=3.4, plotsim=FALSE, type="l")
Qsim <- superPos(valid$P, UH)
Qsim <- Qsim + valid$Q[1] # add baseflow
lines(Qsim, lwd=2, xpd=NA)
legend("center", legend=c("Observed","Simulated from calibration"),
       lwd=c(1,2), col=c(2,1) )
nse(valid$Q, Qsim[1:nrow(valid)]) # 0.47, which is not really good.
# performs OK for the first event, but misses the peak from the second.
# this particular UH is apparently not suitable for high pre-event soil moisture.
# Along with longer events, UH properties may change!!!
dummy # in-sample NSE 0.75 is a lot better

# Now for the second peak in the validation dataset:
lsc(valid$P, valid$Q, type="l", area=3.4, fit=60:90) # overestimates first peak
# Area cannot be right - is supposedly 17 km^2.

## Not run in Rcmd check after Version 1.5 because it takes so much time
## Not run:

# Different starting points for optim:
lsc(calib$P, calib$Q, area=3.4, n= 2  , k=  3, plot=FALSE) # Default
lsc(calib$P, calib$Q, area=3.4, n= 5  , k= 20, plot=FALSE) # same result
lsc(calib$P, calib$Q, area=3.4, n=10  , k= 20, plot=FALSE) # ditto
lsc(calib$P, calib$Q, area=3.4, n=10  , k=  3, plot=FALSE) # ditto
lsc(calib$P, calib$Q, area=3.4, n= 1.9, k=900, plot=FALSE) # ditto
lsc(calib$P, calib$Q, area=3.4, n=50  , k= 20) # nonsense
# the catchment is small, so n must be low.


#sensitivity against area uncertainty:
Asens <- data.frame(A=seq(1,15,0.5),
    t(sapply(seq(1,15,0.5), function(A) lsc(calib$P, calib$Q, area=A, plot=FALSE))))
Asens
plot(Asens$A, Asens$NSE, type="l", ylim=c(-0.3,2), las=1, main="lsc depends on area")
abline(v=3.4, lty=2)
lines(Asens$A, Asens$n, col=2)
points(3.4, 2, col=2)
lines(Asens$A, Asens$psi, col=5)
text(rep(13,4),y=c(1.5, 0.8, 0.4,0), c("k ->","<- NSE","<- n","<- psi"), col=c(4,1,2,5))
par(new=TRUE); plot(Asens$A, Asens$k, type="l", ann=FALSE, axes=FALSE, col=4)
axis(4, col.axis=4)
points(3.4, 3, col=4)

# Autsch - that shouldn't happen!
# Still need to find out what to do with optim


lsc(calib$P, calib$Q, area=1.6) # not bad indeed

## End(Not run)
```

`lsMem` *Show memory size of objects in MB*

Description

Show memory size of the biggest objects in MB. Helps you find the biggest memory killers.

Usage

```
lsMem(n = 6, pos = 1, ...)
```

Arguments

`n`	Number of Objects to be shown separately. The rest is combined into "sum rest". DEFAULT: 6
`pos`	Environment where `ls` looks for objects.
`...`	Further arguments passed to `ls`

Value

Named vector with object sizes in MB (MegaBytes)

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2014

References

http://stackoverflow.com/questions/1358003/tricks-to-manage-the-available-memory-in-an-r-session

See Also

`object.size`, `ls`

Examples

```
## Not run:
## excluded from CRAN check  - I forgot why, but there's probably a good reason
lsMem()

## End(Not run)
```

monthAxis *Label date axis*

Description

Labels date axes at sensible intervals in the time domain of weeks to decades.

Usage

```
monthAxis(side = 1, timeAxis = NA, origin = "1970-01-01",
  startyear = NULL, stopyear = NULL, n = 5, npm = NULL, npy = NA,
  format = "%d.%m.\n%Y", labels = format.Date(d, format), ym = FALSE,
  mcex = 0.6, mmgp = c(3, 0, 0), midyear = FALSE, midmonth = FALSE,
  midargs = NULL, mgp = c(3, 1.5, 0), cex.axis = 1, tick = TRUE,
  tcl = par("tcl"), las = 1, ...)
```

Arguments

side	Which axis are to be labeled? (can be several). DEFAULT: 1
timeAxis	Logical indicating whether the axis is POSIXct, not date. DEFAULT: NA, meaning axis value >1e5
origin	Origin foras.Date and as.POSIXct. DEFAULT: "1970-01-01"
startyear	Integer. starting year. DEFAULT: NULL = internally computed from par("usr")
stopyear	Ditto for ending year. DEFAULT: NULL
n	Approximate number of labels that should be printed (as in pretty). DEFAULT: 5
npm	Number of labels per month, overrides n. DEFAULT: NULL = internally computed.
npy	Number of labels per year, overrides npm and n. DEFAULT: NA
format	Format of date, see details in strptime. DEFAULT: "%d.%m.\n%Y"
labels	labels. DEFAULT: format.Date(d, format)
ym	Label months with first letter at the center of the month and year at center below. Sets midyear and midmonth to TRUE. Uses labels and format for the years, but ignores them for the months. DEFAULT: FALSE
mcex	cex.axis for month labels if ym=TRUE. DEFAULT: 0.8
mmgp	mgp for month labels if ym=TRUE. DEFAULT: 3,0,0
midyear	Place labels in the middle of the year? if TRUE, format default is "%Y". DEFAULT: FALSE
midmonth	Place labels in the middle of the month? if TRUE, format default is "%m\n%Y". DEFAULT: FALSE
midargs	List of arguments passed to axis for the year-start lines without labels. DEFAULT: NULL

mgp	MarGinPlacement, see par. The second value is for label distance to axis. DEFAULT: c(3,1.5,0)
cex.axis	CharacterEXpansion (letter size). DEFAULT: 1
tick	Draw tick lines? DEFAULT: TRUE
tcl	Tick length (negative to go below axis) in text line height units like mgp[2] Changed to -2.5 for year borders if ym=TRUE. DEFAULT: par("tcl")
las	LabelAxisStyle for orientation of labels. DEFAULT: 1 (upright)
...	Further arguments passed to axis, like lwd, col.ticks, hadj, lty, ...

Value

The dates that were labelled

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2015, update labels and midyear Dec 2015

See Also

monthLabs for the numbercrunching itself, axis.Date with defaults that are less nice.

Examples

```
set.seed(007) # for reproducibility
Date1 <- as.Date("2013-09-25")+sort(sample(0:150, 30))
plot(Date1, cumsum(rnorm(30)), type="l", xaxt="n", ann=FALSE)
monthAxis(side=1)
monthAxis(1, npm=2, cex.axis=0.5, col.axis="red") # fix number of labels per month

DateYM <- as.Date("2013-04-25")+0:500
plot(DateYM, cumsum(rnorm(501)), type="l", xaxt="n", ann=FALSE)
monthAxis(ym=TRUE)
monthAxis(ym=TRUE, mgp=c(3,1,0))
monthAxis(ym=TRUE, cex.axis=1.4)
monthAxis(ym=TRUE, mcex=0.9, col.axis="red")

plot(Date1, cumsum(rnorm(30)), type="l", xaxt="n", ann=FALSE)
monthAxis(labels=FALSE, col.ticks=2)
monthAxis(1, format=" ")  # equivalent to axis(labels=FALSE)
monthAxis(1)
d <- monthAxis(1, labels=letters[1:24], mgp=c(3,2.5,0))
d # d covers the full year, thus is longer than n=5

Date2 <- as.Date("2011-07-13")+sort(sample(0:1400, 50))
plot(Date2, cumsum(rnorm(50)), type="l", xaxt="n", ann=FALSE)
monthAxis(npy=12, format=" ")  # fix number of labels per year
monthAxis(tcl=-0.8, lwd.ticks=2, format="%Y/%m", mgp=c(3,1,0))
monthAxis(format="", mgp=c(3,2,0)) # International Date format YYYY-mm-dd
```

```
plot(Date2, cumsum(rnorm(50)), type="l", xaxt="n", ann=FALSE)
monthAxis(midyear=TRUE)
abline(v=monthLabs(npm=1), col=8)

Date3 <- as.Date("2011-07-13")+sort(sample(0:1200, 50))
plot(Date3, cumsum(rnorm(50)), type="l", xaxt="n", ann=FALSE)
monthAxis(1, n=4, font=2)
monthAxis(1, col.axis=3) # too many labels with default n=5

# mid-year labels:
plot(Date3, cumsum(rnorm(50)), type="l", xaxt="n", ann=FALSE)
monthAxis(midyear=TRUE, midargs=list(tcl=-1.2))

# mid-month labels:
plot(Date1, cumsum(rnorm(30)), type="l", xaxt="n", ann=FALSE)
monthAxis(midmonth=TRUE)

# Time axis instead of date axis:
plot(as.POSIXct(Sys.time()+c(0,10)*24*3600), 1:2, xaxt="n")
monthAxis(n=3)
monthAxis()
```

`monthLabs` *Nicely spaced labels along a month*

Description

Create dates of certain days of the month for labeling

Usage

```
monthLabs(startyear = 2002, stopyear = 2018, npm = 2, npy = NA)
```

Arguments

`startyear`	Integer. starting year. DEFAULT: 2002
`stopyear`	Integer. ending year. DEFAULT: 2018
`npm`	Integer, one of 1,2,3,6 or 31. Number of labels per month. DEFAULT: 2 npm : days of the month 1 : first day of each month within the given years 2 : 1st and 15th day 3 : 1, 10, 20 6 : 1, 5, 10, 15, 20, 25. 31 : each day
`npy`	Integer, one of 1,2,3,4 or 6. Number of labels per year at equally spaced month-beginnings. If specified, npm is not considered at all. DEFAULT: NA

Value

Vector with Dates as returned by as.Date.

Note

Spacing of days is not equal, but set to dertain days of the month! This was originally developed for time series movie frames

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, early 2013

See Also

monthAxis for automatic determination of npm/npy, as.Date, paste

Examples

```
monthLabs(2014,2014, 3) # 3 days per month
monthLabs(2013,2014,  npy=3) # 3 months per year, equally spaced
monthLabs(2014,2014,  npy=4) # 4 months per year

# see monthAxis for automatic plot labelling
```

---

movAv *Moving average*

Description

Weighted moving average (running mean) with overlapping windows

Usage

```
movAv(dat, width = 7, weights = rep(1, width), quiet = FALSE)
```

Arguments

dat	Vector with regularly spaced data
width	Odd integer specifying window width. DEFAULT: 7
weights	Vector with weights. Sum is normalized to 1. DEFAULT: rep(1,width)
quiet	Logical: suppress allNA message and even width warning? DEFAULT: FALSE

Details

Width has to be odd, so there is a defined middle point of each window. Even inputs will be changed with a warning (unless quiet=TRUE).
Weights doesn't have to be symmetrical, but is always mapped to the middle of each window!
If there are NAs in the window, the corresponding weight is distributed evenly to the other weights.

Value

Vector of the same length as the original input. Padded with NAs at width/2 margin elements

Note

You can specify just one of weights or width.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, ca 2012

See Also

filter, decompose, smooth, loess, rollapply (no overlapping!)

Examples

```
# general usage ------------------------------------------------------------
set.seed(29); a <- runif(40, 5,50)
data.frame(a, movAv(a))

# final and commencing NAs are kept, middle ones are filled:
a[c(1:10, 18:26, 32:40)] <- NA
data.frame(a, movAv(a))

set.seed(29); a <- runif(60, 5,50)
plot(a, type="o", pch=16, las=1)
lines(movAv(a), col=2, lwd=3) # shows trends, signal in the noise
lines(movAv(a,3), col=4, lwd=3)
lines(movAv(a,15), col=3, lwd=3) # degree of smoothing depends on window width

# Weights:
plot(a, type="o", pch=16, las=1)
lines(movAv(a), col=2, lwd=3) # uniform weight within running window
# Triangular weights react stronger to extrema:
lines(movAv(a, weights=c(1,2,4,6,4,2,1)), col=4, lwd=3)

plot(c(Nile), type="l")
lines(movAv(Nile,20), col=4, lwd=4)
lines(movAv(Nile,21), col=3) # even widths are changed to a higher value


# smoothing intenstiy ------------------------------------------------------
plot(1871:1970, Nile, type="l", col=8)
```

```
movAvLines(1871:1970, Nile, lwd=3)

for(i in 1:30*2-1)
  {
  plot(a, type="o", pch=16, las=1, main=paste("moving average, width =", i))
  lines(movAv(a, i), col=2, lwd=4)
  }
# How to lie with moving averages: compare width 29 with 49 - the "trend"
# appears to be in opposite direction! (OK, this is random data anyways).

b <- rep(a, each=10)+runif(600, -10, 20)
plot(b, type="l")
lines(movAv(b), col=2, lwd=4)
lines(movAv(b, 35), col=4, lwd=4)
lines(movAv(b, 101), col=5, lwd=4)  # choose width according to scale!


# Deviance from running mean can identify outlier:
nile <- c(Nile)
op <- par(mfrow=c(3,1), mar=c(1,3,2.5,0), cex.main=1, las=1)
plot(nile, type="l", main=c("original Nile data",""), xlab="", xaxt="n")
lines(movAv(nile,5), lwd=2, col=2)
title(main=c("", "5-element running mean (moving average)"), col.main=2)
box("figure")
plot(nile-movAv(nile,5), type="o", pch=16, col=4,
      main="difference  ( original data - moving average )", xlab="", xaxt="n")
abline(h=0)
box("figure")
par(mar=c(3,3,1,0))
hist(nile-movAv(nile,5), breaks=25, xlim=c(-500,500), col=4, main="Deviances")
abline(v=0, lwd=5) # the deviances are pretty symmetric.
# If this were shifted more strongly to the left, we could say:
# movav(5) overestimates minima more than it underestimates maxima
# This would happen if low values peak away further and more shortly
par(op)
```

`movAvLines` *Moving average with different window widths*

Description

Add moving average lines with different window widths to a plot

Usage

```
movAvLines(x = 1:length(y), y, widths = c(3, 5, 7, 9, 11, 13), weights,
  col = "blue", alpha = 0.3, add = TRUE, las = 1, ...)
```

Arguments

x	x values of data. DEFAULT: 1:length(y)
y	y values that are smoothed with several window widths
widths	widths of movAv windows. DEFAULT: 2:7*2-1
weights	weights within each window
col	color passed to addAlpha. DEFAULT: "blue"
alpha	transparency passed to addAlpha. DEFAULT: 0.3
add	Logical: Add to existing plot?Set to FALSE to first create the scatterplot. DEFAULT: TRUE
las	LabelAxisStyle (only relevant if add=FALSE). DEFAULT: 1
...	further arguments passed to lines

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2015

See Also

movAv, addAlpha

Examples

```
set.seed(42)
movAvLines(y=cumsum(rnorm(50)), add=FALSE, lwd=3)
```

---

mReg *Multiple regression*

Description

Multiple regression fitting various function types including e.g. linear, cubic, logarithmic, exponential, power, reciprocal. Quick way to find out what function type fits the data best. Plots data and fitted functions and adds a legend with the functions (or their types=structure) sorted by R squared. Returns the fitted functions with their parameters and R^2 values in a data.frame.

Usage

```
mReg(x, y = NULL, data = NULL, Poly45 = FALSE, exp_4 = FALSE,
  xf = deparse(substitute(x)), yf = deparse(substitute(y)), ncolumns = 9,
  plot = TRUE, add = FALSE, nbest = 12, R2min, selection = NULL,
  digits = 2, extend = 0.4, xlim = extendrange(x, f = extend),
  ylim = extendrange(y, f = extend), xlab = xf, ylab = yf, las = 1,
  lwd = rep(1, 12), lty = rep(1, 12), col = NULL, pcol = par("col"),
  pch = 16, legend = TRUE, legargs = NULL, legendform = "nameform",
  quiet = FALSE, ...)
```

Arguments

x	Vector with x coordinates or formula (like y~x), the latter is passed to model.frame
y	Vector with y values. DEFAULT: NULL (to enable x to be a formula)
data	data.frame in which formula is applied. DEFAULT: NULL
Poly45	Logical. Should 4th and 5th degree polynomials also be fitted? DEFAULT: FALSE, as the formulas are very long.
exp_4	Logical. Return 4-parametric exponential distribution fits (via exp4p) in the output table? (only best fit is plotted). exp_4par_ini has the initial values of exponential fitting with the data relocated to first quadrant. The others are optimized with the methods of optim. DEFAULT: FALSE
xf	Character. x name for Formula. DEFAULT: substitute(x) before replacing zeros in x and y
yf	Ditto for y
ncolumns	Number of columns in output. Set lower to avoid overcrowding the console. DEFAULT: 9
plot	Logical. plot data and fitted functions? DEFAULT: TRUE
add	Logical. add lines to existing plot? DEFAULT: FALSE
nbest	Integer. Number of best fitting functions to be plotted (console output table always has all). DEFAULT: 12
R2min	Numerical. Minimum Rsquared value for function type to be plotted. Suggestion: 0.6 (2/3 of variation of y is explained by function of x). DEFAULT: empty
selection	Integers of functions to be plotted, assigned as in list in section "note". DEFAULT: NULL, meaning all
digits	Integer. number of significant digits used for rounding formula parameters and R^2 displayed. DEFAULT: 2
extend	Numerical. Extention of axis ranges (proportion of range). DEFAULT: 0.4
xlim	Numerical vector with two values, defining the x-range of the lines to be plotted. DEFAULT: extended range(x)
ylim	Ditto for Y-axis
xlab	Character. default labels for axis labeling and for formulas. DEFAULT: substitute(x) before replacing zeros in x and y
ylab	Ditto for y axis.
las	Integer in 0:4. label axis style. See par. DEFAULT: 1
lwd	Numerical of length 12. line width for lines. DEFAULT: rep(1,12)
lty	Numerical of length 12. line type. DEFAULT: rep(1,12)
col	Numerical of length 12. line colors. DEFAULT: NULL, means they are specified internally
pcol	Color used for the data-points themselves. DEFAULT: par('col')
pch	Integer or single character. Point CHaracter for the data points. See par. DEFAULT: 16

legend	Logical. Add legend to plot? DEFAULT: TRUE
legargs	List. List of arguments passed to legend. Will overwrite internal defaults. DEFAULT: NULL
legendform	One of 'full', 'form', 'nameform' or 'name'. Complexity (and length) of legend in plot. See Details. DEFAULT: 'nameform'
quiet	Suppress warnings about value removal (NAs, smaller 0, etc)? DEFAULT: FALSE
...	Further graphical parameters passed to plot

Details

legendform : example
full : 7.8*x + 6.31
form : a*x+b
nameform : linear a*x+b
name : linear

full can be quite long, especially with Poly45=TRUE!

Value

data.frame with rounded R squared, formulas, and full R^2 and parameters for further use. Rownames are the names (types) of function. Sorted decreasingly by R^2

warning

A well fitting function does NOT imply correct causation!
A good fit does NOT mean that you describe the bahaviour of a system adequatly!
Extrapolation can be DANGEROUS!
Always extrapolate to see if a function fits the expected results there as well.
Avoid overfitting: Poly45 will often yield good results (in terms of R^2), but can be way overfitted. And outside the range of values, they act wildly.

Note

If you're adjusting the appearance (lwd, lty, col) of single lines, set parameters in the following order:
1 linear a*x + b
2 quadratic (parabola) a*x^2 + b*x + c
3 kubic a*x^3 + b*x^2 + c*x + d
4 Polynom 4th degree a*x^4 + b*x^3 + c*x^2 + d*x + e
5 Polynom 5 a*x^5 + b*x^4 + c*x^3 + d*x^2 + e*x + f
6 logarithmic a*log(x) + b
7 exponential a*e^(b*x)
8 power/root a*x^b
9 reciprocal a/x + b
10 rational 1 / (a*x + b)

11 exponential 4 Param a*e^(b*(x+c)) + d

Negative values are not used for regressions containing logarithms; with warning.
exp_4par was originally developed for exponential temperature decline in a cup of hot water.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2012, updated April and Aug 2013, sept 2015

References

Listed here: http://rclickhandbuch.wordpress.com/rpackages

See Also

glm, lm, optim

Examples

```
set.seed(12)
x <- c(runif(100,0,3), runif(200, 3, 25)) # random from uniform distribution
y <- 12.367*log10(x)+7.603+rnorm(300)     # random from normal distribution
plot(x,y, xlim=c(0,40))
mReg(x,y) # warning comes from negative y-values (suppress with quiet=TRUE)

# Formula specification:
mReg(Volume~Height, data=trees)

# NA management
x[3:20] <- NA
mReg(x,y)

# Passing arguments to legend:
mReg(x,y, pch=1, legargs=list(x="bottomright", cex=0.7), legendform="form")

mReg(x,y, col=rainbow2(11))
mReg(x,y, extend=0.2) # less empty space around data points
mReg(x,y, nbest=4) # only 4 distributions plotted
mReg(x,y, legargs=list(x=7, y=8, bty="o", cex=0.6)) # Legend position as coordinates

## Not run: # Excluded from Rcmd check (opening external devices)
View(mReg(x,y, Poly45=TRUE, exp_4=TRUE, plot=FALSE)) # exp_4: fit more distributions

## End(Not run)
# optim methods often yield different results, so be careful using this.
# I might insert a possibility to specify initial values for optim.
# 4 Parameters allow several combinations to yield similarly good results!
plot( 0:10, 3.5*exp(0.8*( 0:10 + 2      )) + 15 , type="l")
lines(0:10,  18*exp(0.8*( 0:10 - 2.5e-05)) - 5, col=2)
```

```
# okay, different dataset:
x <- c(1.3, 1.6, 2.1, 2.9, 4.4, 5.7, 6.6, 8.3, 8.6, 9.5)
y <- c(8.6, 7.9, 6.6, 5.6, 4.3, 3.7, 3.2, 2.5, 2.5, 2.2)
mReg(x,y, legargs=list(cex=0.7, x="topright"), main="dangers of extrapolation")
points(x,y, cex=2, lwd=2)
# Polynomial fits are good within the data range, but, in this case obviously,
# be really careful extrapolating! If you know that further data will also be low,
# add another point to test differences:
mReg(c(x,11,13,15), c(y,2,2,2), xf="myX", yf="myY", Poly45=TRUE, legendform="name")
points(x,y, cex=2, lwd=2)
# The Polynomials are still very good: they have 5 to 6 Parameters, after all!
# Poly45 is set to FALSE by default to avoid such overfitting.

mReg(x,y, pcol=8, ncol=0) # no return to console

# only plot a subset: best n fits, minimum fit quality, or user selection
mReg(x,y, pcol=8, ncol=2, nbest=4)
mReg(x,y, pcol=8, ncol=2, R2min=0.7)
mReg(x,y, pcol=8, ncol=2, selection=c(2,5,8))
# selecting the fifth degree polynomial activates Poly45 (in the output table)

# Add to axisting plot:
plot(x,y, xlim=c(0,40))
mReg(x,y, add=TRUE, lwd=12:1/2, ncol=0)
# lwd, lty can be vectors of length 12, specifying each line separately.
# Give those in fix order (see section notes), not in best-fit order of the legend.
# The order is Polynomial(1:5), log, exp, power, reciprocal, rational, exp_4_param
# color has to be a vector of 12
# opposedly, lwd and lty are repeated 12 times, if only one value is given


# One more dataset:
j <- c(5,8,10,9,13,6,2) ; k <- c(567,543,587,601,596,533,512)
# Inset from margin of plot region:
mReg(j,k, legargs=list(x="bottomright", inset=.05, bty="o"), legendform="name")
# Legend forms
mReg(j,k, legargs=list(x="bottomright"), legendform="name")
mReg(j,k, legargs=list(x="bottomright"), legendform="form")
mReg(j,k, legargs=list(x="bottomright"), legendform="nameform")
mReg(j,k, legargs=list(x="bottomright"), legendform="full")

## Not run: # Excluded from Rcmd check (long computing time)

# The question that got me started on this whole function...
# exponential decline of temperature of a mug of hot chocolate
tfile <- system.file("extdata/Temp.txt", package="berryFunctions")
temp <- read.table(tfile, header=TRUE, dec=",")
head(temp)
plot(temp)
temp <- temp[-20,] # missing value - rmse would complain about it

x <- temp$Minuten
y <- temp$Temp
```

```
mReg(x,y, exp_4=TRUE, selection=11)
# y=49*e^(-0.031*(x - 0  )) + 25 correct, judged from the model:
# Temp=T0 - Te *exp(k*t) + Te     with    T0=73.76,  Tend=26.21, k=-0.031
# optmethod="Nelder-Mead"  # y=52*e^(-0.031*(x + 3.4)) + 26 wrong


x <- seq(1, 1000, 1)
y <- (x+22)/(x+123) # can't find an analytical solution so far. Want to check out nls
mReg(x, y, legargs=list(x="right"))


## End(Not run)


# Solitaire Results. According to en.wikipedia.org/wiki/Klondike_(solitaire):
# Points=700000/Time + Score
# I recorded my results as an excuse to play this game a lot.
sfile <- system.file("extdata/solitaire.txt", package="berryFunctions")
solitaire <- read.table(sfile, header=TRUE)
mReg(solitaire$Time, solitaire$Points) # and yes, reciprocal ranks highest! Play Fast!
mReg(solitaire$Time, solitaire$Bonus, xlim=c(50,200), extend=0, nbest=3)
sol <- unique(na.omit(solitaire[c("Time","Bonus")]))
sol
sol$official <- round(700000/sol$Time/5)*5
mReg(sol$Time, sol$Bonus, extend=0, selection=9, col=rep(4,10), legendform="full")
plot(sol$Time, sol$official-sol$Bonus, type="l")

# multivariate regression should be added, too:
sfile <- system.file("extdata/gelman_equation_search.txt", package="berryFunctions")
mv <- read.table(sfile, header=TRUE)

sfile <- system.file("extdata/mRegProblem.txt", package="berryFunctions")
x <- read.table(sfile, header=TRUE)$x
y <- read.table(sfile, header=TRUE)$y
mReg(x,y,  digits=6) # all very equal
x2 <- x-min(x)
mReg(x2,y, digits=6)          #  Formulas are wrong if digits is too low!!
#mReg(x2,y, legendform="full")

# Zero and NA testing (to be moved to unit testing someday...)
mReg(1:10, rep(0,10))
mReg(1:10, c(rep(0,9),NA))
mReg(1:10, rep(NA,10))
mReg(rep(1,10), 1:10)
mReg(rep(0,10), 1:10)
mReg(c(rep(0,9),NA), 1:10)
mReg(rep(NA,10), 1:10)

mReg(1:10, rep(0,10), quiet=TRUE)
mReg(1:10, c(rep(0,9),NA), quiet=TRUE)
mReg(1:10, rep(NA,10), quiet=TRUE)
mReg(rep(1,10), 1:10, quiet=TRUE)
mReg(rep(0,10), 1:10, quiet=TRUE)
```

```
mReg(c(rep(0,9),NA), 1:10, quiet=TRUE)
mReg(rep(NA,10), 1:10, quiet=TRUE)
```

---

na9 *Prepend spaces before na.strings*

Description

Returns a number of useful character strings with varying amount of spaces prepended. It can be used as `na.strings=na9()` in `read.table`.

Usage

```
na9(nspace = 5, base = c(-9999, -999, -9.99, -9.999), sep = c(",", "."),
  digits = 0:4, more = NULL, ...)
```

Arguments

nspace	number of spaces prepended. DEFAULT: 5
base	Numeric: basic na.string numbers
sep	Separator string (comma or decimal point or both). DEFAULT: c(",",".")
digits	Number(s) of zeros to be appended. DEFAULT: 0:4
more	More structures added to base, like "NA", "–". digits and sep is not added to this! DEFAULT: NULL
...	Arguments passed to nothing currently

Value

Character strings

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2016

See Also

`paste`

Examples

```
na9()
na9(nspace=0, sep=".")
na9(nspace=0, sep=".", more=c(NA,"-"))
```

nameSample *Nonrandom character sequence with sample*

Description

Find the seed necessary to produce a character sequence by using sample

Usage

```
nameSample(name, progress = FALSE, estimatetime = nc > 4,
  continue = FALSE)
```

Arguments

name	Character string. long strings (»5) will compute a VERY long time!
progress	Logical. Monitor progress by printing a dot every 10000 tries? DEFAULT: TRUE for long names (nchar(name)>3).
estimatetime	Estimate computation time? DEFAULT: nc>4
continue	Continue without asking? DEFAULT: FALSE

Value

cats command into the console that can be copypasted to anyone's R script.

Note

nameSample may take a lot of time, due to nchar^26 possibilities. That's why it warns about strings longer than 5 characters

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, April 2014

See Also

yearSample to wish a happy new year, set.seed, sample, letters

Examples

```
## Not run in RCMD check as they're very time consuming
## Not run:
nameSample("berry")  # After that, you can send the result to colleagues:
# Kind regards from
set.seed(1248272); paste(sample(letters,5,TRUE), collapse='')

# calculation time
                                      # on my slow laptop:   # on PC
```

```
system.time(nameSample("berr"))        # 25 s # berry: 57 s     10  23
system.time(nameSample("berr", FALSE)) # 23 s          53 s      9  20

# let <- sapply(1:4, function(n) apply(replicate(n, letters[sample(15)]), 1, paste, collapse=""))
# calctime <- sapply(let, function(x) system.time(nameSample(x, progress=F))[3])
# write.table(calctime, "calctime_nameSample.txt")
ctfile  <- system.file("extdata/calctime_nameSample.txt",  package="berryFunctions")
ctfile2 <- system.file("extdata/calctime_nameSample2.txt", package="berryFunctions")
calctime <- read.table(ctfile)
# regression result in hours:
expReg(nchar(rownames(calctime))-8, calctime[,1], xlim=c(1,7), ylim=c(-3,4),
       predict=7)/3600

# For my 3 times faster computer:
calctime <- read.table(ctfile2)
expReg(nchar(rownames(calctime))-8, calctime[,1], xlim=c(1,7), ylim=c(-3,4),
       predict=c(4,7))/c(1,3600)
# 4 sec for 4 letters are expected to be 10 hours for 7 letters...


## End(Not run)
```

`newFilename` *Create new filename if file already exists*

Description

Check if files already exist and append _1 or _2, etc to the filename if needed, thereby giving useful messages.

Usage

```
newFilename(filename, ignore = FALSE, pre = "", mid = "\n  ", end = "",
  quiet = FALSE, ntrunc = 3)
```

Arguments

`filename`	Char (vector): file name(s).
`ignore`	Logical (vector, recycled): Ignore file? DEFAULT: FALSE
`pre, mid, end`	Char: strings to append after traceback / message / filenames. DEFAULT: "", "\n ", ""
`quiet`	Logical: Suppress messages about creating file(s)? DEFAULT: FALSE
`ntrunc`	Integer: Number of filenames printed in messages before they get truncated with message "(and xx more)". DEFAULT: 3

Value

newFilename returns the input with an added "_n" in the filename for each file that already existed.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Oct 2016 + Jan 2017

See Also

file.exists

Examples

```
fns <- c("data", "stupiddummy", "ExampleGraph.png", "berryFunctions.Rproj",
         "README.md", "stupiddummy.txtdude", "DESCRIPTION", "test_devel.R")
newFilename(fns)
newFilename(fns, ignore=TRUE)
newFilename(fns, ignore=rep(0:1, each=4))
newFilename(fns, ntrunc=2)
newFilename("README.md")
newFilename("dummy", mid="") # no line break
```

---

normPlot *Normal density plot*

Description

Nice plot of normal density distribution

Usage

```
normPlot(mean = 0, sd = 1, width = 3, lines = TRUE, quant = TRUE,
  fill = addAlpha("blue", c(2:6, 7:2)/10), cumulative = TRUE, las = 1,
  main = paste("Normal density with\nmean =", signif(mean, 2), "and sd =",
  signif(sd, 2)), ylim = lim0(dnorm(mean, mean, sd)), ylab = "",
  xlab = "", type = "n", lty = 1, col = par("fg"), mar = c(2, 3, 3,
  3), keeppar = FALSE, ...)
```

Arguments

mean	average value as in dnorm. DEFAULT: 0
sd	standard deviation. DEFAULT: 1
width	distance (in sd) from plot ends to mean. DEFAULT: 3
lines	Should vertical lines be plotted at mean +- n*sd? DEFAULT: TRUE
quant	should quantile regions be drawn with fill colors? DEFAULT: TRUE
fill	color(s) passed to polygon. DEFAULT: addAlpha("blue",c(2:6,7:2)/10)
cumulative	Should cumulative density distribution be added? DEFAULT: TRUE
las	arguments passed to plot. DEFAULT: 1

main	main as in plot. DEFAULT: paste("Normal density with\nmean =", mean, "and sd =", sd)
ylim	limit for the y axis. DEFAULT: lim0(y)
ylab, xlab	labels for the axes. DEFAULT: ""
type, lty, col	arguments passed to lines. type="l" to add pdf line
mar	margins for plot passed to par. DEFAULT: c(2,3,3,3)
keeppar	should margin parameters be kept instead of being restored to previous value? DEFAULT: FALSE
...	further arguments passed to plot like lwd, xaxs, cex.axis, etc.

Details

This function finds some nice defaults for very quickly plotting a normal distribution by just specifying mean and sd.

Value

None. Used for plotting.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2014

See Also

betaPlot, dnorm, https://cran.r-project.org/package=denstrip, https://cran.r-project.org/view=Distributions

Examples

```
normPlot()
normPlot(81.7, 11.45)
normPlot(180, 11, quant=FALSE, width=2)
```

---

owa *Overwrite argument default lists*

Description

Second ellipsis (three dots) passed to particular functions, combining default and user-specified argument lists.
owa can be used in functions that pass argument lists separately to several functions. Internal defaults can be set per function (eg. one list for plot and one for legend).
You can specify which defaults can be overwritten and which should be left unchanged. See the example section on how to implement this.

Usage

```
owa(d, a, ..., quiet = FALSE)
```

Arguments

d	Default arguments (list or vector)
a	Arguments specified by user (list or vector)
...	Names of unchangeable arguments (that will not be overwritten) as character strings. Can also be a vector with characters strings.
quiet	Logical: Should message be suppressed if arguments are ignored? If FALSE (the DEFAULT), this helps users debugging, as they get notified when arguments they specified were ignored.

Value

Always a list, disregarding list/vector mode of input

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Early 2014, Update Oct 2016

References

http://stackoverflow.com/questions/3057341
http://stackoverflow.com/questions/5890576
http://stackoverflow.com/questions/4124900
http://stackoverflow.com/questions/16774946

Examples

```
# The motivation behind owa:
testfun <- function(...) {plot(7:11, ...) ; legend("top", "some text", ...)}
testfun()
is.error( testfun(type="o") , tell=TRUE)
# Error: legend doesn't have the argument 'type'!

# How to solve this:
testfun <- function(legargs=NULL, ...) # dots passed to plot
   {
   plot(7:11, ...)
   legend_defaults <- list(x="top", lty=1, col="red", legend="owa rocks!")
   # combine defaults and user specified into final argument list,
   # overwrite arguments ('owa') in the default list unless protected:
   legend_final <- owa(d=legend_defaults, a=legargs, "col", "lwd")
   do.call(legend, args=legend_final)
   }

testfun()
testfun(type="l", col="blue")
```

```
testfun(type="o", legargs=list(col="blue", pch=16, lty=3) )
# color in legargs is ignored, as it is defined as unchangeable


#------------------------------------------------------------------------

# basic tests of owa itself:
d <- list(bb=1:5, lwd="was d", lty=1,   col="gray")
a <- list(bb=3,   lwd=5, lty="from a", wachs="A")
owa(d,a) # all changed, wachs added
owa(d, a, "bb", "lwd") # lty is overwritten, bb and lwd are ignored
owa(d, NULL, "bb", "wachs") # NULL is a good default for argument lists
owa(d, c(HH=2, BBB=3) ) # vectors and lists are all converted to lists
owa(d, list(lwd=5, bb=3, lty="1") ) # order of arguments doesn't matter
owa(d, a, c("bb","lwd") ) # unchangeable can also be a named vector
owa(d, a, c("bb","lwd"), c("lty","dummy") ) # or several vectors
```

`panelDim` *Arrange panels in a multipanel plot (par mfrow)*

Description

Returns the optimum where deviation from ncol=nrow and number of panels left empty have a minimum sum.

Usage

```
panelDim(n, weight = c(1, 1), maxempty = round(n/4), landscape = FALSE,
  all = FALSE, plot = FALSE, mfcol = FALSE)
```

Arguments

`n`	Number of panels to be arranged
`weight`	Weights to avoid *empty panels* and *discrepancy between ncol and nrow*, respectively. DEFAULT: c(1,1)
`maxempty`	Maximum number of panels that are allowed to be left empty. If `maxempty=0`, no panel is left blank, so 11 plots would be beneath each other instead of in a 4x3 grid with one panel left blank. DEFAULT: round(n/4)
`landscape`	Use landscape orientation instead of portrait? DEFAULT: FALSE
`all`	Show all reasonable possibilities in a data.frame? DEFAULT: FALSE
`plot`	Show the panel layout result? (the 4 best options are compared if `all=TRUE`). DEFAULT: FALSE
`mfcol`	use mfcol instead of mfrow. DEFAULT: FALSE

Details

There probably are other ways to find the optimal way to arrange panels, so if you find anything, please give me a hint.

Value

vector with 2 values, can be passed to par(mfrow), or a data.frame if `all=TRUE`.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2014, Jan 2015

See Also

`groupHist`, which is using this function

Examples

```
# basic usage
op <- par(mfrow=panelDim(6))
for(i in 1:6) plot(i:10, main=i)
par(op)

# Adcanced options
panelDim(7)
g <- panelDim(7, all=TRUE)
panelDim(7, plot=TRUE)
panelDim(7, plot=TRUE, all=TRUE) # compares 4 best options

panelDim(26, all=TRUE)
panelDim(26, plot=TRUE, all=TRUE) # compares 4 best options
panelDim(26, plot=TRUE, all=TRUE, weight=c(3,0) ) # fewer empty panels

# effect of maxempty:
panelDim(13, plot=TRUE)             #  4 x 4
panelDim(13, maxempty=2, plot=TRUE) #  5 x 3
panelDim(13, maxempty=1, plot=TRUE) #  7 x 2
panelDim(13, maxempty=0, plot=TRUE) # 13 x 1

panelDim(45, plot=TRUE) # no empty panels
# focus on aspect ratio of each panel (make it as square as possible):
panelDim(45, weight=c(1,3), plot=TRUE) # better aspect for each panel

# Orientation of plot:
panelDim(45, plot=TRUE) # good for portrait orientation of plot
panelDim(45, landscape=TRUE, plot=TRUE) # better if plot width > height

## Not run:
## Rcmd check --as-cran doesn't like to open external devices,
## so this example is excluded from running in the checks.
plot of several n with defaults
```

```
dev.new(record=TRUE)
for(i in 1:50)  panelDim(i, plot=TRUE)

## End(Not run)
```

---

pdfpng *Create pdf and png graph*

Description

Create both a pdf and a png file with a graph, with custom size default values. To iteratively create pdfs without closing and reopening the pdf viewer, you might want to use e.g. Sumatra, which comes with Rstudio. It can be found e.g. in C:/Program Files/RStudio/bin/sumatra

Usage

```
pdfpng(expr, file, pdf = TRUE, png = TRUE, overwrite = FALSE,
  quiet = FALSE, filargs = NULL, width = 7, height = 5, units = "in",
  res = 500, seed = runif(1, -1e+09, 1e+09), envlevel = 1,
  pdfargs = NULL, pngargs = NULL, ...)
```

Arguments

expr	Expression creating the plot, can be included in curly braces.
file	Character: Filename without pdf/png extension. Unless overwrite=TRUE, files will not be overwritten, but "_1" will be appended instead, see newFilename. If expr creates several plots, use file="fname otherwise the png will only contain the last figure. Note: this overwrites files as the captured by newFilename. You may also have to run dev.off().
pdf	Logical: Create pdf? DEFAULT: TRUE
png	Logical: Create png? DEFAULT: TRUE
overwrite	Logical: Overwrite existing file? Can be a vector for pdf and png separately. DEFAULT: FALSE (_n appended in filename)
quiet	Logical: suppress file creation messages? DEFAULT: FALSE
filargs	List of other arguments passed to newFilename. DEFAULT: NULL
width, height	Graph dimensions. DEFAULT: 7x5 inches
units, res	Graph quality arguments passed only to png. DEFAULT: inches ("in"), 500 ppi
seed	Seed passed to set.seed before each call. DEFAULT: runif(1,-1e9,1e9)
envlevel	Environment level passed to eval.parent. Never needs to be changed, as far as I can tell. DEFAULT: 1
pdfargs	List of arguments only passed to pdf.
pngargs	List of arguments only passed to png.
...	Further arguments passed to both pdf and png

Value

Nothing

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, March 2017

See Also

pdf, png

Examples

```
pdfpng({par(bg=8, las=1); plot(cumsum(rnorm(500)), type="l")},
       file="dummyplot", res=100)
pdfpng({par(bg=8, las=1); plot(cumsum(rnorm(500)), type="l")},
       file="dummyplot", overwrite=c(TRUE,FALSE))

# Nesting of functions is possible:
a <- list( cumsum(rnorm(2000)), cumsum(rnorm(20)) )
pdfpng(plot(a[[1]]), file="dummyplot", overwrite=TRUE)
bfun <- function(b) pdfpng(plot(b,type="l"), file="dummyplot", overwrite=TRUE)
cfun <- function(c) bfun(c)
bfun(a[[1]])
sapply(a, function(d) cfun(d))


unlink("dummyplot.pdf") ; unlink("dummyplot.png") ; unlink("dummyplot_1.png")
```

---

popleaf *create leaflet popup box info*

Description

combine data.frame columns into a leaflet popup-box compatible format

Usage

```
popleaf(df, sel = colnames(df))
```

Arguments

df	Data.frame
sel	Columns to be selected (Names or index or TRUE/FALSE vector). DEFAULT: colnames(df)

Value

Vector with character strings

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Apr 2017

See Also

paste

Examples

```
dat <- data.frame(a=14:16, b=letters[14:16], c=LETTERS[14:16],
                  lat=c(52.58,53.45,52.4), lon=c(6.34,7.23,13.05))
popleaf(dat)
dat$display <- popleaf(dat, 1:4)

## Not run:  # Excluded from CRAN checks
library(leaflet)
leaflet(dat) %>% addTiles() %>% addCircleMarkers(~lon, ~lat, popup=~display)

## End(Not run)
```

---

pretty2 *Truncated pretty breakpoints*

Description

pretty with no values outside of x range

Usage

```
pretty2(x, n = 5, force = FALSE, ...)
```

Arguments

x	object with numeric values
n	desired number of values in pretty. DEFAULT: 5
force	Must output lenght equal n exactly? DEFAULT: FALSE
...	all other arguments in pretty.

Details

calculates pretty(x), then removes the values that do not lie within range(x).
If force=TRUE, range(x) is reduced step by step in a while loop until the condition is met. This is useful if you want exactly 2 labels on an axis. In order not to get stuck, the outer values are taken if there are more than n values within range(x).

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Aug 2014

See Also

pretty , logVals

Examples

```
k <- c(135, 155, 120, 105, 140, 130, 190, 110)
range(k)
pretty(k)
pretty2(k)

pretty(c(0.2, 0.9), n=2)
pretty2(c(0.2, 0.9), n=2)
pretty2(c(0.2, 0.9), n=2, force=TRUE)
```

---

quantileBands *Quantile bands*

Description

Quantile bands with optional smoothing, e.g. for visualizing simulations

Usage

```
quantileBands(mat, x = 1:ncol(mat), col = rgb(0, 0, 1, alpha = c(0.5, 0.7)),
  add = FALSE, main = "Quantile Bands", ylab = "", xlab = "",
  probs = 0:4/4, na.rm = FALSE, type = 7, smooth = NA, medargs = NULL,
  meanargs = NULL, txi, textargs = NULL, ...)
```

Arguments

mat	Matrix or data.frame with columns of data
x	X-axis positions for each column. DEFAULT: 1:ncol(mat)
col	Vector of colors for each quantile group, recycled reversively if necessary. DEFAULT: rgb(0,0,1, alpha=c(0.5, 0.7))
add	Add to existing plot? Allows to add to highly customized plot. DEFAULT: FALSE
main, xlab, ylab	plot labels. DEFAULT: "Quantile Bands", ""
probs	Probabilities passed to quantile. DEFAULT: 0:4/4
na.rm	Remove NAs before computing quantiles, median and mean? DEFAULT: FALSE

type	Which of the 9 quantile algorithms should be used. DEFAULT: 7
smooth	If(!is.na), width passed to movAv smoothing quantiles. DEFAULT: NA
medargs	List of arguments passed to lines drawing median. Not drawn if NULL. DEFAULT: NULL
meanargs	List of arguments passed to lines drawing mean. Not drawn if NULL. DEFAULT: NULL
txi	Text x position index (along columns of mat), recyled if necessary. NA to suppress. INTERNAL DEFAULT: middle of the plot for all.
textargs	List of arguments passed to text, like col, adj, ... DEFAULT: NULL
...	Further arguments passed to polygon, like border, lty, ...

Value

Quantiles of each column, invisible. Smoothed if smooth is given!

Note

This is the first version and is not tested very well yet.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

See Also

quantile, quantileMean, ciBand, polygon, https://cran.r-project.org/package=fanplot

Examples

```
neff <- t(replicate(n=30, sapply(1:400, function(nn) max(rnorm(nn)))   ))
qB <- quantileBands(neff, x=1:400)
qB[,1:9]
quantileBands(neff, smooth=19, meanargs=list(col=2), txi=NA)

library(RColorBrewer)

quantileBands(neff, smooth=35, ylab="max of rnorm(n)",
  xlab="sample size (n)", probs=0:10/10, col=brewer.pal(5,"BuGn"),
  medargs=list(lwd=2), meanargs=list(col=2, lty=1), txi=c(40,50,60),
  main="Maximum is an unsaturated statistic:\n it rises with sample size")

neff2 <- t(replicate(n=50, sapply(1:400, function(nn) mean(rnorm(nn)))   ))
quantileBands(neff2, x=1:400, smooth=35, ylab="mean of rnorm(n)",
  xlab="sample size (n)", probs=0:10/10, col=brewer.pal(5,"BuGn"),
  txi=c(40,50,60), textargs=list(col="yellow"), medargs=list(lwd=2),
  meanargs=list(col=2, lty=1), main="Mean converges to true population mean")
```

quantileMean *Average of R's quantile methods*

Description

Weighted average of R's quantile methods

Usage

```
quantileMean(x, probs = seq(0, 1, 0.25), weights = rep(1, 9),
  names = TRUE, truncate = 0, ...)
```

Arguments

x	Numeric vector whose sample quantiles are wanted
probs	Numeric vector of probabilities with values in [0,1]. DEFAULT: seq(0, 1, 0.25)
weights	Numeric vetor of length 9 with weight for each quantile method. Recycled if shorter. DEFAULT: unweighted mean. DEFAULT: rep(1,9)
names	If TRUE, the resulting vector has a names attribute. DEFAULT: TRUE
truncate	Number between 0 and 1. Censored quantile: fit to highest values only (truncate lower proportion of x). Probabilities are adjusted accordingly. DEFAULT: 0
...	further arguments passed to quantile, except for type

Details

weights are internally normalized to sum 1

Value

numeric named vector, as returned by apply

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

See Also

quantile

Examples

```
exDat <- rnorm(30,sd=5)
quantile(exDat, probs=c(0.9, 0.99), type=1)
quantile(exDat, probs=c(0.9, 0.99), type=2)
round( sapply(1:9, function(m) quantile(exDat, probs=0.9, type=m)) , 3)
# and now the unweighted average:
```

```
quantileMean(exDat, probs=c(0.9, 0.99))
quantileMean(exDat, probs=0.9)
# say I trust type 2 and 3 especially and want to add a touch of 7:
quantileMean(exDat, probs=c(0.9, 0.99), weights=c(1,5,5,0,1,1,3,1,1))

# quantile sample size dependency simulation:
qbeta(p=0.999, 2, 9) # dist with Q99.9% = 0.62
betaPlot(2, 9, cumulative=FALSE)
abline(v=qbeta(p=0.999, 2, 9), col=6, lwd=3)
qm <- function(size) quantileMean(rbeta(size, 2,9), probs=0.999, names=FALSE)
n30  <- replicate(n=500, expr=qm(30))
n1000 <- replicate(n=500, expr=qm(1000))
lines(density(n30)) # with small sample size, high quantiles are systematically
lines(density(n1000), col=3) # underestimated. for Q0.999, n must be > 1000


## Not run:
# #Excluded from CRAN Checks because of the long computing time
# median of 500 simulations:
qmm <- function(size, truncate=0) median(replicate(n=500,
      expr=quantileMean(rbeta(size, 2,9), probs=0.999, names=FALSE, truncate=truncate)))

n <- seq(10, 1000, length=30)
medians <- sapply(n, qmm)  # medians of regular quantile average
plot(n, medians, type="l", las=1)
abline(h=qbeta(p=0.999, 2, 9), col=6) # real value
# with truncation:
medians_trunc <- sapply(n, qmm, truncate=0.8) # only top 20% used for quantile estimation
lines(n, medians_trunc, col=2) # censored quantiles don't help!
# In small samples, rare high values do not occur on average

# Parametrical quantiles can avoid sample size dependency!
if(!require(devtools)) install.packages("devtools")
devtools::install_github("brry/extremeStat")
library("extremeStat")
library2("pbapply")

distLquantile(rbeta(1000, 2,9), probs=0.999, plot=TRUE, nbest=10) # 10 distribution functions
distLquantile(rbeta(1000, 2,9), probs=0.999, plot=TRUE, nbest=10) # that seem to work well
select <- c("wei","wak","pe3","ln3","kap","gno","gev","gum","gpa","gam")

pqmm <- function(size, truncate=0, plot=FALSE) median(replicate(n=50,
       expr=mean(distLquantile(rbeta(size, 2,9), probs=0.999, type=select,
          plot=plot, nbest=10, progbars=FALSE, time=FALSE, truncate=truncate))))

#dev.new(record=TRUE)
#pqmm(30, plot=TRUE)

# medians of parametrical quantile estimation
###suppressMessages(pmedians <- pbsapply(n, pqmm) )  # takes several minutes
write.table(pmedians, file="../inst/extdata/pmedians.txt", row.names=FALSE, col.names=FALSE)
pmedians <- read.table("../inst/extdata/pmedians.txt")[,1]
```

```
plot(n, medians, type="l", ylim=c(0.4, 0.7), las=1)
abline(h=qbeta(p=0.999, 2, 9), col=6) # real value
lines(n, medians_trunc, col=2) # censored quantiles don't help!
lines(n, pmedians, col=4) # overestimated, but not dependent on n
# with truncation, only top 20% used for quantile estimation
suppressMessages(pmedians_trunc <- pbsapply(n[-1], pqmm, truncate=0.8))
lines(n[-1], pmedians_trunc, col=6) # much better!
# Good for this beta distribution. I don't know how it scales to other dists.

## End(Not run)
```

`rainbow2` *Rainbow from blue to red*

Description

Reversed `rainbow` with different defaults, resulting in a color vector from blue (good) to red (bad)

Usage

```
rainbow2(n = 10, s = 1, v = 1, start = 0, end = 0.7, alpha = 1)
```

Arguments

`n`	number of colors. DEFAULT: 10
`s, v`	saturation and value as in `rainbow`. DEFAULT: 1
`start`	start color. DEFAULT: 0
`end`	end color. DEFAULT: 0.7
`alpha`	transparency. DEFAULT: 1)

Value

A character vector of color names.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

See Also

`seqPal` for a better palette, `rainbow`

Examples

```
plot(1:10, pch=16, cex=2, col=rainbow2(10))
```

`removeSpace` *Remove white spaces from strings*

Description

Remove leading and/or trailing white space from character strings

Usage

```
removeSpace(x, begin = TRUE, end = TRUE, all = FALSE, ...)
```

Arguments

`x`	Character string, can be a vector
`begin`	Logical. Remove leading spaces at the beginning of the character string? DEFAULT: TRUE
`end`	Logical. Remove trailing spaces at the end? DEFAULT: TRUE
`all`	Logical. Remove all spaces anywhere in the string? DEFAULT: FALSE
`...`	Further arguments passed to sub or gsub, like `ignore.case, perl, fixed, useBytes`.

Value

Character string (vector)

Note

If all arguments are FALSE, the string is returned unchanged.
Not extensively tested yet, please mail me any problems...

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2014

See Also

sub

Examples

```
s <- c("space at end     ", "  white at begin", "  both ", " special ^  ")
removeSpace(s)

# To add space, use:
x <- c("ab","abcde")
format(x)
format(x, justify="centre")
format(x, width=9)
```

`rescale` *shift and scale a vector*

Description

rescale a numeric vector: map values linearly onto a given range

Usage

```
rescale(x, from = 0, to = 1)
```

Arguments

`x`	Numerical vector of values to be mapped to a given range
`from`	output minimum. DEFAULT: 0
`to`	output maximum. DEFAULT: 1

Value

numeric vector, rescaled onto output range

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2016

References

http://stackoverflow.com/a/18303620

See Also

`scales::rescale`

Examples

```
rescale(10:15, 135, 200)
rescale(10:15, 200, 135)
rescale(10:15, to=c(1,5))

values <- rbeta(1e3, shape1=4, shape2=35)
hist(rescale(values, 135, 200), breaks=25, col=3)
```

runAxis *Label axis with typical running times*

Description

Label a numerical axis (in minutes) with time units that are typical for running times (10 sec intervals)

Usage

```
runAxis(t = 3 * 60, int1 = 10, int2 = 5, side = 1, linarg = NULL, ...)
```

Arguments

t	Maximum time in minutes
int1	Primary interval (for labels)
int2	Secondary interval (for lines)
side	Side of the plot to draw axis (1,2,3,4 = bottom, left, top, right)
linarg	List of arguments passed to abline
...	Further arguments passed to axis

Value

List with the positions and labels

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jun 2016

See Also

logAxis, monthAxis

Examples

```
plot(1:200, xaxt="n")
runAxis(t=200, int1=20, int2=10)
```

seasonality *Seasonality analysis*

Description

Examine time series for seasonality of high (low) values

Usage

```
seasonality(dates, values, data, drange = NA, vrange = NA, shift = 0,
  janline = TRUE, nmax = 0, maxargs = NULL, plot = 1, add = FALSE,
  nmin = 100, probs = c(0, 25, 50, 75, 95, 99.9)/100, width = 3,
  text = TRUE, texti = seq(200, 20, length.out = length(probs)),
  textargs = NULL, months = substr(month.abb, 1, 1), slab = "Month",
  tlab = "Year", vlab = NA, xlim = NA, ylim = NA, xaxs = NA,
  yaxs = NA, main = "Seasonality", adj = 0.2, mar = c(3, 3, 4, 1),
  mgp = c(1.7, 0.7, 0), keeppar = TRUE, legend = TRUE, legargs = NULL,
  returnall = FALSE, quiet = FALSE, ...)
```

Arguments

dates	Dates in ascending order. Can be charater strings or strptime results, as accepted (and coerced) by as.Date
values	Values to be mapped in color with colPoints
data	Optional: data.frame with the column names as given by dates and values
drange	Optional date range (analogous to xlim), can be a vector like dates. DEFAULT: NA (computed from dates internally)
vrange	Optional value range (analogous to ylim), can be a vector like values. DEFAULT: NA (computed from values internally)
shift	Number of days to move the year-break to. E.g. shift=61 for German hydrological year (Nov to Oct). DEFAULT: 0
janline	Logical: Should horizontal line be plotted at January 1st if shift!=0? DEFAULT: TRUE
nmax	Number of annual maxima to be marked, plotted and returned. Currently, only 0 and 1 are implemented. DEFAULT: 0
maxargs	List of arguments passed to lines for annual maxima, e.g. maxargs=list(type="l", col="red", lty= DEFAULT: NULL (several internal defaults are used, but can be overridden)
plot	Integer specifying the type of plot. Can be a vector to produce several plots. 0: none, only return the data.frame with annual maxima. 1: color coded doy (day of the year) over year (the default). 2: Color coded spiral graph with spiralDate. 3: Spaghetti line plot with discharge over doy, one line per year. 4: probs quantileMean over doy, with optional aggregation window (width) around each doy. 5: Annmax over time for crude trend analysis. DEFAULT: 1

add	Logical. Add to existing plot? DEFAULT: FALSE
nmin	Minimum number of values that must be present per (hydrological) year to be plotted in plot type 5. DEFAULT: 100
probs	Probabilities passed to quantileMean for plot=4. DEFAULT: c(0,25,50,75,95,99)/100
width	Numeric: window width for plot=4. Used as sd in gaussian weighting. Support (number of values around a DOY passed to quantile funtion at least once) is ca 4.9*width. The value at doy itself is used 10 times. Larger values of width require more computing time. DEFAULT: 3
text	Logical. Call textField if plot=4? DEFAULT: TRUE
texti	Numerical (vector): indices at which to label the lines. DEFAULT: seq(200,20,length.out=length(probs))
textargs	List of arguments passed to textField for plot=4. DEFAULT: NULL
months	Labels for the months. DEFAULT: J,F,M,A,M,J,J,A,S,O,N,D
slab, tlab, vlab	Labels for the **s**eason, **t**ime (year) and **v**alues used on the axes and title of colPointsLegend. DEFAULT: "Month", "Year", substitute(values)
xlim, ylim	Limits of x and y axis. DEFAULT: NA (specified internally per plot type)
xaxs, yaxs	x and y Axis style, see par. Use "r" for regular 4% expansion, "i" for in range only. DEFAULT: NA (specified internally per plot type)
main, adj	Graph title and offset to the left (adj passsed to title). DEFAULT: "Seasonality", 0.2
mar, mgp	Parameters specifying plot margin size and labels placement. DEFAULT: c(3,3,4,1), c(1.7,0.7,0) (Changed for plot 3:5 if not given)
keeppar	Logical: Keep the margin parameters? If FALSE, they are reset to the previous values. DEFAULT: TRUE
legend	Logical. Should a legend be drawn? DEFAULT: TRUE
legargs	List of arguments passed as legargs to colPoints. DEFAULT: NULL (internally, plots 3 and 5 have density=F as default)
returnall	Logical: return all relevant output as a list instead of only annmax data.frame? DEFAULT: FALSE
quiet	Logical: suppress progress stuff and colPoints messages? DEFAULT: FALSE
...	Further arguments passed to colPoints like pch, main, xaxs, but not Range (use vrange). Passed to spiralDate if plot=2, like add, format, lines.

Value

The output is always invisible, don't forget to assign it. If returnall=FALSE: Data.frame with year, number of nonNA entries, max value + doy of annual maxima. Please note that the column year does not match the calendrical year if shift!=0.
if returnall=TRUE: a list with annmax (df from above) as well as:
data: data.frame(doy, values, year) and optionally:
plot1, plot3, plot4, plot5: outputs from colPoints
plot2: output list from spiralDate
and other elements depending on plot type, like data3, data4, probs4, width4.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jul-Oct 2016

See Also

spiralDate, colPoints, https://waterdata.usgs.gov/nwis

Examples

```
# browseURL("http://nrfa.ceh.ac.uk/data/station/meanflow/39072")
qfile <- system.file("extdata/discharge39072.csv", package="berryFunctions")
Q <- read.table(qfile, skip=19, header=TRUE, sep=",", fill=TRUE)[,1:2]
rm(qfile)
colnames(Q) <- c("date","discharge")
Q$date <- as.Date(Q$date)
Q$discharge[450:581] <- NA
plot(Q, type="l")
seas <- seasonality(date, discharge, data=Q, shift=100, main="NRFA: Thames\nRoyal Windsor Park")
head(seas)
# notice how n for nonmissing values is lower in one single hydrological year,
# which includes parts of two consecutive calendarical years.

# Be careful with your interpretation. This looks normal up to 2007, but then BAM!:
seasonality(date, discharge, data=Q[Q$date<as.Date("2007-07-15"),], plot=3, shift=100, nmax=1)
seasonality(date, discharge, data=Q[Q$date<as.Date("2007-08-15"),], plot=3, shift=100, nmax=1)

# Shift is important. You don't want to have this event included twice:
seasonality(date, discharge, data=Q[850:950,], plot=3, nmax=1, quiet=TRUE, shift=100)

seasonality(date, discharge, data=Q, plot=2) # most floods in winter
seasonality(date, discharge, data=Q, plot=5, vlab="Dude, look at annual max Q!")
seasonality(date, discharge, data=Q, plot=5, shift=100)
s <- seasonality(date, discharge, data=Q, plot=4, shift=100, width=3, returnall=TRUE)
str(s, max.lev=1)

## Not run:  # excluded from CRAN checks because it is slow
seasonality(date, discharge, data=Q, plot=3:4, add=0:1, ylim=lim0(400), shift=117)
seasonality(date, discharge, data=Q, plot=4, add=TRUE, lwd=3, shift=117, width=3)

## End(Not run)

## Not run:
dev.new(noRStudioGD=TRUE, record=TRUE)     # large graph on 2nd monitor
par(mfrow=c(2,2))
seasonality(date, discharge, data=Q, plot=(1:5)[-4], shift=100)
seasonality(date, discharge, data=Q, plot=(1:5)[-4], lwd=2)
seasonality(date, discharge, data=Q, plot=(1:5)[-4], nmax=1, shift=100)
seasonality(date, discharge, data=Q, plot=(1:5)[-4], col=divPal(100, ryb=TRUE))
dev.off()

## End(Not run)
```

seqPal *Sequential color palette*

Description

Sequential color palette from yelow to red or yellow to blue or custom colors.

Usage

```
seqPal(n = 12, reverse = FALSE, alpha = 1, extr = FALSE, yb = FALSE,
  yr = FALSE, gb = FALSE, colors = NULL, logbase = 1, ...)
```

Arguments

n	Number of colors. DEFAULT: 12
reverse	Reverse colors? DEFAULT: FALSE
alpha	Transparency (0=transparent, 1=fully colored). DEFAULT: 1
extr	Should colors span possible range more extremely? If TRUE, it has very light yellow and very dark blue values included, using the result from RColorBrewer::brewer.pal(9, "YlGnB DEFAULT: FALSE
yb	Should colors be in yellow-blue instead of the internal (nice) default? DEFAULT: FALSE
yr	Should colors be in yellow-red instead of the default? DEFAULT: FALSE
gb	Should colors be in green-blue instead of the default? DEFAULT: FALSE
colors	If not NULL, a color vector used in colorRampPalette. DEFAULT: NULL
logbase	If !=1, this is passed to classify and logSpaced. DEFAULT: 1
...	Further arguments passed to colorRamp

Value

Character string vector with color names

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jan 2016

See Also

showPal, divPal, addAlpha, colorRampPalette, package RColorBrewer

Examples

```
plot(rep(1,12),  pch=16, cex=5, col=seqPal(12), xaxt="n")
showPal()

# nonlinear color scale (use colPoints + see classify for more options):
v <- rescale(volcano^30)
image(v, col=seqPal(1000), asp=1);  colPointsLegend(v, nbins=1000)
image(v, col=seqPal(1000, logbase=1.007), asp=1)
colPointsLegend(v, col=seqPal(1000, logbase=1.09))

plot(   rep(1, 1000), pch=15, cex=3, col=seqPal(1000), ylim=c(0.99, 1.01), ylab="logbase", las=1)
for(b in seq(0.99, 1.01, len=30))
    points(rep(b, 1000), pch=15, cex=1, col=seqPal(1000, logbase=b))
```

---

seqR *seq with a range argument*

Description

sequence given by range or vector of values.

Usage

```
seqR(range, from = 1, to = 1, extend = 0, ...)
```

Arguments

range	vector with 2 values (1st taken as from, 2nd as to) or more (the result is then always ascending).
from	start value of sequence. DEFAULT:1
to	end value of sequence. DEFAULT:1
extend	Factor f passed to extendrange. DEFAULT:0
...	further arguments passed to seq.

Value

Numeric vector.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Feb 2014

See Also

seq, range, http://r.789695.n4.nabble.com/seq-range-argument-td4684627.html

Examples

```
seqR(range=c(12,6), by=-2)
m <- c(41, 12, 38, 29, 50, 39, 22)
seqR(m, len=6)
# Takes min and max of range if the vector has more than two elements.

seqR(range=c(12,6), by=-2, extend=0.1)
# internaly calls extendrange with f=extend
```

---

showPal *show color palettes*

Description

Plot examples of the seqential and diverging color palettes in this package. Do not use rainbow: https://eagereyes.org/basics/rainbow-color-map

Usage

```
showPal(cex = 4, ...)
```

Arguments

cex	Caracter EXpansion size (width of color bar). DEFAULT: 4
...	Arguments passed to par

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Apr 2016

See Also

seqPal, divPal, package RColorBrewer

Examples

```
showPal()
```

`smallPlot` *Inset small plot within figure*

Description

multipanel-compatible inset plot with margins, background and border

Usage

```
smallPlot(expr, x1 = 0.05, x2 = 0.7, y1 = 0.5, y2 = 1, outer = FALSE,
  xpd = NA, mar = c(3, 3, 1, 1), mgp = c(1.8, 0.8, 0), bg = par("bg"),
  border = par("fg"), las = 1, resetfocus = TRUE, colwise = FALSE, ...)
```

Arguments

`expr`	expression creating a plot. Can be code within braces.
`x1, x2, y1, y2`	Position of small plot, relative to current figure region [0:1]. DEFAULT: x: 0.05-0.7, y: 0.5-1
`outer`	Logical. Should inset plot be placed in the device outer margin region instead of relative to the current figure region? Useful in multipanel plots with par(oma). `outer` here does not have exactly the same meaning as in `title`. DEFAULT: FALSE
`xpd`	Plotting and notation clipped to plot region (if xpd=FALSE), figure region (TRUE) or device region (xpd=NA). DEFAULT: NA
`mar`	Margin vector in (approximate) number of lines. It is internally multiplied with `strheight` to convert it to relative units [0:1], thus the behaviour is a bit different from `par(mar)`. It's recycled, so you can use `mar=0`. DEFAULT: c(3,3,1,1)
`mgp`	MarGinPlacement: distance of xlab/ylab, numbers and line from plot margin, as in `par`, but with different defaults. DEFAULT: c(1.8, 0.8, 0)
`bg`	Background. DEFAULT: par("bg")
`border`	Border around inset plot. DEFAULT: par("fg")
`las`	LabelAxisStyle. DEFAULT: 1
`resetfocus`	Reset focus to original plot? Specifies where further low level plot commands are directed to. DEFAULT: TRUE
`colwise`	Logical: Continue next plot below current plot? If you had `par(mfcol=...)`, you must use `colwise=TRUE`, otherwise the next plot will be to the right of the current plot (as with `par(mfrow=...)`). DEFAULT: FALSE
`...`	further arguments passed to `par`. This may mess things up - please tell me for which arguments! You can do `par(las=1, las=2)` (the last will be set), so `smallPlot(plot(1), new=FALSE)` works, but may not yield the intended result.

Value

parameters of small plot, invisible.

Warning

setting mai etc does not work!

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2014-2016

See Also

colPointsLegend for an example of usage. subplot and add.scatter for alternative solutions to this problem that do not set margins.

Examples

```
# Basic usage:
op <- par(no.readonly=TRUE) # original parameters
plot(1:10)
smallPlot(plot(5:1, ylab="Yo man!"), bg="lightgreen" )
smallPlot(plot(5:1), x1=0.5,x2=1, y1=0.3,y2=0.6, bg="yellow", yaxt="n")
# if R warns "figure margins too large", try dragging the plot viewer bigger

# select focus for further add-on's:
points(2, 2, pch="+", cex=2, col=2) # main window
smallPlot( plot(5:1), bg="lightblue", resetfocus=FALSE )
mtext("dude")
points(2, 2, pch="+", cex=2, col=2) # smallPlot window
par(op)

# More par settings:
plot(1:10)
smallPlot( plot(50:1), bg=6, mai=c(0.2, 0.3, 0.1, 0.1)) # screws up
smallPlot( plot(5:1), bg=8, ann=FALSE)
smallPlot(plot(10:50), bg="transparent") # old plot is kept

# complex graphics in code chunks:
plot(1:100)
smallPlot( {plot(5:1, ylab="Rocky label"); lines(c(2,4,3));
            legend("topright", "BerryRocks!", lwd=3)    }, bg="white")


# multiple figure situations
par(op)
par(mfcol=c(3,4))
plot(1:10)
plot(1:10)
smallPlot(plot(5:1), bg="lightblue")
plot(1:10)
smallPlot(plot(5:1), bg="bisque", colwise=TRUE) # if mfcol (not mfrow) was set
plot(1:10)

# Outer margins (e.g. to add legends to multi-panel plots)
```

```
par(op)
par(mfrow=c(3,2), oma=c(0,5,0,0), mar=c(0,0,1,0)+0.5)
for(i in 0:5*4) image(volcano+i, zlim=c(90,200), xaxt="n", yaxt="n",
                      main=paste("volcano +", i))
smallPlot(plot(1:10), x1=0,x2=0.25, y1=0.5,y2=1, bg="green", mar=1)
smallPlot(plot(1:10), x1=0,x2=0.25, y1=0.5,y2=1, bg="green", mar=1, outer=TRUE)
colPointsLegend(90:200, horizontal=FALSE, x1=0, col=heat.colors(12), outer=TRUE,
               labelpos=5, density=FALSE, title="", cex=2, lines=FALSE)


# Further testing with mfrow and mfcol
par(op)
old_plt <- par("plt")
par(mfcol=c(3,4))
new_plt <- par("plt")
plot(1:10)
plot(1:10)
smallPlot(plot(5:1), bg="lightblue", colwise=TRUE)
points(3, 2, pch="+", cex=2, col=2)
plot(1:10) # canot keep mfcol, only mfrow, if colwise is left FALSE.
smallPlot(plot(5:1), bg="bisque", resetfocus=FALSE )
points(3, 2, pch="+", cex=2, col=2)
plot(1:10) # in smallPlot space
par(plt=old_plt)
plot(1:10) # too large
smallPlot(plot(5:1), bg="palegreen")
points(3, 2, pch="+", cex=2, col=2, xpd=NA) # not drawn with default xpd
par(plt=new_plt)
plot(1:10) # canot keep mfcol, only mfrow, if colwise is left FALSE.
smallPlot(plot(5:1), bg="yellow")
points(3, 2, pch="+", cex=2, col=2)   # everything back to normal
par(op)

# if layout is used instead of par(mfrow), it is difficult to add graphs
# after using smallPlot
lay <- matrix(c(1,1,1,1,2,2,3,3,2,2,3,3,4,4,5,5), ncol=4)
layout.show(layout(lay))
layout(lay)
plot(1:10)
plot(1:10)
smallPlot(plot(1:10), mar=c(1,3,1,0), x1=0,x2=0.2, y1=0.2,y2=0.8, bg=4, outer=TRUE)
# plot(1:10) # now in a weird location (par("mfrow") is 4x4 after layout)
```

`smoothLines` *draw smoothed lines*

Description

draw smoothed lines with an n-level partially transparent haze

Usage

```
smoothLines(x, y, lwd = 1, col = 1, n = 5, alpha = 0.1, ...)
```

Arguments

x	numrical. x-coordinates. x can be a matrix, then the y coordinates are taken from the second column
y	numerical. y-coordinates
lwd	single integer. line width
col	color. DEFAULT: 1 (black)
n	single integer. number of transparent lines overlayed with sinking line widths. DEFAULT: 5
alpha	Transparency of color. DEFAULT: 0.1 (very transparent)
...	further arguments as in lines

Value

none, draws lines

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011/2012

See Also

lines, col2rgb, rgb

Examples

```
x <- 1:5  ; y <- c(0.31, 0.45, 0.84, 0.43, 0.25)
plot(x,y)
smoothLines(x,y)
#png("smoothLines.png")
par(mar=c(2,2,2,0)+.5)
plot(1:100, las=1, type="n", main="usage of blines(x,y, lwd, col, n, alpha ...)")
abline(h=0:10*10, v=0:10*10, col=6); box()
for(i in 0:9) { smoothLines(x=c(0,10,25,35), y=c(i*10, i*10, i*10+12, i*10+7), lwd=i)
                text(25, i*10+5, paste("n=5,lwd=", i, sep="")) }
for(i in 0:9) { smoothLines(x=c(40,50,65,75), y=c(i*10, i*10, i*10+12, i*10+7), n=i)
                text(65, i*10+5, paste("n=",i,",lwd=1", sep="")) }
for(i in 0:9/20) { smoothLines(x=c(80,90,105), y=c(i*200, i*200+12, i*200+12), alpha=i)
                   text(90, i*200+10, paste("alpha=", i, sep=""), adj=0) }
text(5,10, "default", adj=c(0.5,-0.2)); text(45,50, "default", adj=c(0.5,-0.2))

#dev.off()
```

sortDF *sort dataframes by column*

Description

sort a data.frame by column - basically just a wrapper for order

Usage

```
sortDF(df, col, decreasing = TRUE, ...)
```

Arguments

df	Data.frame to be sorted
col	Column (index or (un)qouted name) to be sorted by
decreasing	Logical: should highest value be on top? DEFAULT: TRUE (unlike order!)
...	Further arguments passed to order, like eg na.last or method

Value

data.frame

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, June 2015

See Also

sort, order

Examples

```
sortDF(USArrests[USArrests$Murder>11,], "Assault")
sortDF(USArrests[USArrests$Murder>11,], 3)
```

spiralDate *Spiral graph of time series*

Description

Plot seasonality of (daily) time series along spiral

Usage

```
spiralDate(dates, values, data, drange = NA, vrange = NA,
  months = substr(month.abb, 1, 1), add = FALSE, shift = 0, prop = NA,
  zlab = substitute(values), format = "%Y", nint = 1, ...)
```

Arguments

dates	Dates in ascending order. Can be charater strings or strptime results, as accepted (and coerced) by as.Date
values	Values to be mapped in color with colPoints along seasonal spiral
data	Optional: data.frame with the column names as given by dates and values
drange	Optional date range (analogous to xlim), can be a vector like dates. DEFAULT: NA
vrange	Optional value range (analogous to ylim), can be a vector like values. DEFAULT: NA
months	Labels for the months. DEFAULT: J,F,M,A,M,J,J,A,S,O,N,D
add	Add to existing plot? DEFAULT: FALSE
shift	Number of days to move january 1st clockwise. DEFAULT: 0
prop	Proportion of the data to be actually plotted, used in spiralDateAnim. DEFAULT: NA (ignored)
zlab	Title of colPointsLegend
format	Format of date labels see details in strptime. DEFAULT: "%Y"
nint	Number of interpolation segments between points, only used if lines=TRUE (passed to colPoints). DEFAULT: 1 (with long time series, the colPoints default of 30 is too high!)
...	Further arguments passed to colPoints, but not Range (use vrange)

Value

invisible data.frame with date, vals, and the plotting coordinates

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2016

See Also

seasonality, colPoints, as.Date

Examples

```
# synthetic seasonal Data
set.seed(42)
fakeData <- data.frame(time = as.Date("1985-01-01")+0:5000,
                       vals = cumsum(rnorm(5001))+50          )
fakeData$vals <- fakeData$vals + sin(0:5000/366*2*pi)*max(abs(fakeData$vals))

sp <- spiralDate(time,vals, data=fakeData)
tail(sp)
spiralDate(time,vals, data=fakeData, drange=as.Date(c("1980-01-01", "2004-11-15")), lines=TRUE)

par(mfrow=c(1,3), mar=c(3,3,6,1), mgp=c(2,0.6,0), las=1)
colPoints(time,vals,vals, data=fakeData, col=divPal(100), add=FALSE, legend=FALSE,
          lines=TRUE, pch=NA, nint=1, lwd=2)
title(main="classical time series\nworks badly for long time series\nshows trends well")

seasonality(time, vals, fakeData, col=divPal(100), mar=c(3,3,6,1), legend=FALSE, main="", shift=61)
title(main="yearly time series\nday of year over time\nfails for cyclicity over all year")

spiralDate(time,vals, data=fakeData, col=divPal(100), legargs=list(y1=0.7,y2=0.8))
title(main="spiral graph\nshows cyclic values nicely
            trends are harder to detect\nrecent values = more visual weight")

par(mfrow=c(1,1))

# Data with missing values:
fakeData[1300:1500, 2] <- NA
spiralDate(time,vals, data=fakeData, lines=TRUE) # no problem
# Missing data:
fakeData <- na.omit(fakeData)
spiralDate(time,vals, data=fakeData, lines=TRUE) # problematic for lines
spiralDate(time,vals, data=fakeData, pch=3)      # but not for points

## Real data:
#library2("waterData")
#data(exampleWaterData)
#spiralDate(dates, val, data=q05054000LT, lines=TRUE, lwd=3)
```

---

spiralDateAnim *Animated spiral graph*

Description

Animation of (daily) time series along spiral

Usage

```
spiralDateAnim(dates, values, data, steps = 100, sleep = 0,
  progbar = TRUE, ...)
```

Arguments

dates, values, data	Input as in spiralDate
steps	Number of steps (images) in animation. DEFAULT: 100
sleep	Pause time between frames, in seconds, passed to Sys.sleep. DEFAULT: 0
progbar	Should a progress bar be drawn? Useful if you have a large dataset or many steps. DEFAULT: TRUE
...	Further arguments passed to spiralDate

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, May 2016

See Also

spiralDate, linLogHist

Examples

```
set.seed(42)
x <- as.Date("1985-01-01")+0:5000
y <- cumsum(rnorm(5001))+50
y <- y + sin(0:5000/366*2*pi)*max(abs(y))/2
plot(x,y)

spiralDateAnim(x,y, steps=10, sleep=0.01) # 0.05 might be smoother...
spiralDateAnim(x,y, steps=20)

## Not run:
## Rcmd check --as-cran doesn't like to open external devices such as pdf,
## so this example is excluded from running in the checks.
pdf("spiralDateAnimation.pdf")
spiralDateAnim(x,y, main="Example Transition", col=divPal(100), format=" ")
dev.off()

# if you have FFmpeg installed, you can use the animation package like this:
library2(animation)
saveVideo(spiralDateAnim(x,y, steps=300), video.name="spiral_anim.mp4", interval=0.1,
    ffmpeg="C:/Program Files/R/ffmpeg/bin/ffmpeg.exe")


## End(Not run)
```

superPos *superposition of discharge, unit hydrograph*

Description

superposition of precipitation along unit hydrograph (to simulate Q from P)

Usage

```
superPos(P, UH)
```

Arguments

P	Vector with precipitation values
UH	Vector with discrete values of the Unit Hydrograph. This can be any UH summing to one, not just the storage cascade model.

Value

list with optimized n and k, Nash-Sutcliffe Index, and simulated discharge

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2013

See Also

lsc where superPos is used, unitHydrograph

Examples

```
N <- c(9,5,2,14,1,3) # [mm/hour]
UH <- c(0.1, 0.4, 0.3, 0.1, 0.1) # [1/h]
sum(UH) # sum must be 1

superPos(N, UH)
# If catchment area = 34 km^2 and precipitation is homogenous:
superPos(N/10^3, UH) * 34*10^6 / 3600  # m^3/s  # Add baseflow and you're done...

SP <- data.frame(Prec=c(N, 0,0,0,0),
          P1=c( UH*N[1], 0,0,0,0,0),
          P2=c(0, UH*N[2], 0,0,0,0),
          P3=c(0,0, UH*N[3], 0,0,0),
          P4=c(0,0,0, UH*N[4], 0,0),
          P5=c(0,0,0,0, UH*N[5], 0),
          P6=c(0,0,0,0,0, UH*N[6] ),
          runoff=superPos(N, UH))
SP # SuperPosition
```

```
SPcum <- t( apply(SP[2:7], 1, cumsum) )

plot(N, type="h", col=2:7, lwd=3, xlim=c(1, 10), ylim=c(30,0), lend=1)
par(new=TRUE)
plot(1, type="n", ylim=c(0, 15), xlim=c(1, 10), axes=FALSE, ann=FALSE)
axis(4, las=1)
polygon(x=c(1:10, 10:1), y=c(SPcum[,1], rep(0, 10)), col=2)
for(i in 2:6) polygon(x=c(1:10, 10:1), y=c(SPcum[,i], rev(SPcum[,i-1])), col=i+1)
text(2.5, 1, "Shape of UH")

lines( superPos(N, UH), lwd=3)

plot(UH, type="o", ylim=c(0, 0.4), las=1)
lines(UH, type="h" )


# Effect of distribution of Prec:
P_a <- c(1,2,3,4,5,6,7,8)
P_b <- c(4,4,4,4,4,4,4,4)
P_c <- c(8,7,6,5,4,3,2,1)
sum(P_a) ; sum(P_b) ; sum(P_c)

UH_1 <- unitHydrograph(n=2, k=2.3, t=1:25)
UH_2 <- unitHydrograph(n=5.5, k=1.8, t=1:25)

par(mfrow=c(2,3), mar=c(2,3,2,1), las=1)
plot(P_a, type="h", col=3, lwd=3, ylim=c(0,8), main="Precipitation a")
plot(P_b, type="h", col=4, lwd=3, ylim=c(0,8), main="Precipitation b")
plot(P_c, type="h", col=5, lwd=3, ylim=c(0,8), main="Precipitation c")
#
plot(UH_1, type="l", main="unit hydrograph", ylab="",xlab="Zeit")
lines(UH_2, col=2)
text(c(7,14), c(0.12, 0.07), c("UH_1","UH_2"), col=1:2)
abline(h=0)
#
plot( superPos(P=P_a, UH=UH_1), col=3, ylim=c(0,5), type="l",
      main="Discharge", ylab="Q [m^3/s]")
lines(superPos(P=P_b, UH=UH_1), col=4)
lines(superPos(P=P_c, UH=UH_1), col=5)
legend("topright", c("P a","P b", "P c"), title="with UH_1", col=3:5, lty=1)
#
plot( superPos(P=P_a, UH=UH_2), col=3, ylim=c(0,5), type="l",
      main="Discharge", ylab="Q [m^3/s]")
lines(superPos(P=P_b, UH=UH_2), col=4)
lines(superPos(P=P_c, UH=UH_2), col=5)
legend("topright", c("P a","P b", "P c"), title="with UH_2", col=3:5, lty=1)
```

`tableColVal` *Table with values with value-dependent colored backgrounds in pdf*

Description

Table with numbers and corresponding color in the background of each cell. (heatmap)

Usage

```
tableColVal(mat, main = deparse(substitute(mat)), nameswidth = 0.3,
  namesheight = 0.1, palette = seqPal(100), Range = range(mat, finite =
  TRUE), digits = 2, classargs = NULL, cellargs = NULL, ...)
```

Arguments

mat	Matrix with values and row/column names
main	Title for topleft space. DEFAULT: name of mat object.
nameswidth	Relative width of row names at the left, as a percentage of plot. DEFAULT: 0.3
namesheight	Relative height of column names at the top. DEFAULT: 0.1
palette	Color palette for the heatmap. DEFAULT: seqPal(100)
Range	Range mapped to color palette. DEFAULT: range(mat)
digits	Number of digits rounded to for writing. DEFAULT: 2
classargs	List of arguments specifying how to call classify, e.g. method. DEFAULT: NULL
cellargs	List of arguments passed to text only for the cells.
...	Further arguments passed to all text like cex, col, srt, ...

Details

Create tables with corresponding color in the background of each cell. (heatmap)

Value

List of locations in plot.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Nov 2012 + Nov 2016

See Also

pdf, heatmap

Examples

```
Bsp <- matrix(c(21,23,26,27, 18,24,25,28, 14,17,23,23, 16,19,21,25), ncol=4, byrow=TRUE)
colnames(Bsp) <- paste0("Measure", LETTERS[1:4])
rownames(Bsp) <- paste("prod", 8:11, sep="_")
Bsp
```

```
(  tableColVal(Bsp)   )
tableColVal(Bsp, nameswidth=0.1) # relative to plot width
tableColVal(Bsp, namesheight=0.5, srt=45)

tableColVal(Bsp, cellargs=list(cex=2), col="red")
tableColVal(Bsp, Range=c(10,40))
tableColVal(Bsp, Range=c(20,40))
tableColVal(Bsp, palette=heat.colors(12))
tableColVal(Bsp, palette=c(2,4,7), main="more\nstuff")

Bsp2 <- matrix(rexp(30), ncol=6, byrow=TRUE)
(  tableColVal(Bsp2)  )
tableColVal(Bsp2, digits=4)
colPointsLegend(Bsp2, horizontal=FALSE, x1=0.05, x2=0.15, y1=0.1, y2=0.8, title="")

## Not run:
## Rcmd check --as-cran doesn't like to open external devices such as pdf,
## so this example is excluded from running in the checks.
pdf("TableColVal.pdf", height=5);  tableColVal(Bsp);  dev.off()
system2("open", "TableColVal.pdf")
unlink("TableColVal.pdf")

## End(Not run)
```

`textField` *Write text to plot with halo underneath*

Description

Write text to plot. A field the size of each label is drawn beneath it, so the text can be read easily even if there are many points in the plot. Fields can be rectangular, elliptic or rectangular with roundeed edges.

Usage

```
textField(x, y, labels = seq_along(x), fill = "white", border = NA,
  expression = NA, margin = 0.3, field = "rounded", nv = 1000,
  rounding = 0.75, lty = par("lty"), lwd = par("lwd"), cex = par("cex"),
  xpd = par("xpd"), adj = par("adj"), pos = NULL, offset = 0.5,
  quiet = TRUE, ...)
```

Arguments

`x`	X coordinates, if necessary, they are recycled
`y`	Y coordinates
`labels`	labels to be placed at the coordinates, as in `text`. DEFAULT: seq_along(x)
`fill`	fill is recycled if necessary. With a message when quiet = FALSE. DEFAULT: "white"

border	ditto for border. DEFAULT: NA
expression	If TRUE, labels are converted to expression for better field positioning through expression bounding boxes. If NA, it is set to TRUE for labels without line breaks (Newlines, "\n"). If FALSE, no conversion happens. DEFAULT: NA
margin	added field space around words (multiple of em/ex). DEFAULT: 0.3
field	'rectangle', 'ellipse', or 'rounded', partial matching is performed. DEFAULT: "rounded"
nv	number of vertices for field = "ellipse" or "rounded". low: fast drawing. high: high resolution in vector graphics as pdf possible. DEFAULT: 1000
rounding	between 0 and 1: portion of height that is cut off rounded at edges when field = "rounded". DEFAULT: 0.75
lty	line type. DEFAULT: par("lty")
lwd	line width. DEFAULT: par("lwd")
cex	character expansion. DEFAULT: par("cex")
xpd	expand text outside of plot region ("figure")?. DEFAULT: par("xpd")
adj	vector of length one or two. DEFAULT: par("adj")
pos	in 'text', pos overrides adj values. DEFAULT: NULL
offset	I want the field to still be drawn with adj, but have it based on pos. DEFAULT: 0.5
quiet	Suppress warning when Arguments are recycled? DEFAULT: TRUE
...	further arguments passed to strwidth and text, like font, vfont, family

Details

Specifying pos and offset will currently change the position of the text, but not of the field.
srt is not supported yet.
lend, ljoin and lmitre can not be specified for rect, to keep argument number low.
density (crosshatch etc.) is not supported, as this would distract from the text. # Search Engine Keywords: R Text visible on top R labeling with color underneath R Creating text with a halo R Text with shadow

Value

None

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, April 2013 + March 2014

References

with inspiration taken from ordilabel in package vegan and thanks to Jari Oksanen for his comments

See Also

text; shadowtext in package TeachingDemos, see http://blog.revolutionanalytics.com/2009/05/make-text-stand-out-with-outlines.html, and http://stackoverflow.com/questions/25631216.
s.label in package ade4, which is not so versatile and doesn't work with logarithmic axes

Examples

```
# TextFields with mixed field shapes ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
set.seed(13);  plot(cumsum(rnorm(100)), type="l", main="berryFunctions::textField")
for(i in 2:7) lines(cumsum(rnorm(100)), col=i)
textField(40, 4, "default")
textField(40, 0, "some options", col=2, fill=4, margin=c(-0.4, 0.9), font=2)
# Ellipsis (looks better in vector graphics like pdf):
textField(80, 2, "field='ellipse'", field="ell", mar=c(0.5, 2.3), border=5)
# Rectangular field with edges rounded:
textField(60,-3, "field='Rounded'", field="rounded", fill="orange", cex=1.7)

# Field type can be abbreviated (partial matching), margin may need adjustment:
textField(90, 5, "short", field="ell", fill=7, border=4, mar=-0.4)

# Rounded can also vectorized:
textField(30, c(2,0,-2,-4,-6), paste("rounding =",seq(0,1,len=5)), field="round",
    fill=(2:6), mar=1, rounding=seq(0,1,len=5), border=1)
# turn off warning about recycling:
textField(80, c(-5,-6.5), c("Ja", "Nein"), field="round", fill=6:8, quiet=TRUE)


set.seed(007); plot(rnorm(1e4)) ; abline(v=0:5*2e3, col=8)
# Default settings:
textField(5000, 0, "Here's some good text")
# right-adjusted text (the field box still extends 'margin' stringwidths em):
textField(2000, -1, "Some more (smores!)", cex=1.5, adj=0, col=2)
# Field color, no extra margin beyond baseline (excluding descenders):
textField(2000, -2, "more yet", col=2, fill="blue", margin=0)
# margin can be one number for both x and y direction ... :
textField(1000, 2, "Up we go", fill=7, margin=1.4)
# ... or two (x and y different), even negative:
textField(5000, 2, "to the right", col=2, fill=4, margin=c(-0.4, 0.9))
# Fonts can be set as well:
textField(5000, 1, "And boldly down in bold font", font=2, border=3)
# Text can expand outsinde of the plot region (figure) into the margins:
textField(11000, -2, "Hi, I'm a long block of text", adj=1, fill="red")
textField(11000, -3, "You're not outside the plot!", adj=1, xpd=TRUE, fill="red")
# And most parameters can be vectorized, while x/y are recycled:
textField(3000, c(-3, -3.7), c("0", "good"), border=c("red",3), lty=1:2)


# textField even works on logarithmic axes:
mylabel <- c("This","is (g)","the","ever-\n great","Sparta")
plot(10^runif(5000, -1,2), log="y", col=8)
```

```
textField(1000, c(100,20,4,2,0.5), mylabel, fill=2, mar=0, expression=FALSE)
textField(2500, c(100,20,4,2,0.5), mylabel, fill=4, mar=0, expression=TRUE)
textField(4000, c(100,20,4,2,0.5), mylabel, fill=3, mar=0)
textField(c(1,2.5,4)*1000, 0.2, paste("expression=\n", c("FALSE","TRUE","NA")))

# In most devices, vertical adjustment is slightly off when the character string
# contains no descenders. The default is for centered text:  adj = c(0.5, NA).
# For drawing the field, adj[2] is in this case set to 0.5.
# Text positioning is different for NA than for 0.5, see details of ?text
# I'm working on it through expression, which does not work with newlines yet
```

`TFtest` *Test logical expressions*

Description

Check if logical expressions return what you expect with a truth table

Usage

```
TFtest(..., na = TRUE)
```

Arguments

`...`	Expression(s) with logical operators to be evaluated, with single letters for variables. Each expression is to be separated with a comma
`na`	Logical: should NAs be included in the truth table? DEFAULT: TRUE

Details

This is a nice way to check operator precedence, see `Syntax`

Value

Truth table as data.frame with TRUE and FALSE (and NA) combinations

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Mrz 2016

See Also

`logical`

Examples

```
TFtest(!a & !b)
TFtest(!a & !b, a&b, !(a&b))
TFtest(!a & !b | c)
TFtest(!a & !b | c, na=FALSE)
TFtest(!a)
TFtest(a&b|c, (a&b)|c, a&(b|c), na=FALSE) # AND has precedence over OR
```

---

timer *Timer alarm*

Description

Beeps in a given interval and gives a progress bar in the console

Usage

```
timer(interval = 20, n = 15, write = FALSE)
```

Arguments

interval	alarm interval in seconds. DEFAULT: 20
n	number of alarm signals to be given. DEFAULT: 15
write	Should the actual estimated time be written for overhead computing time control purposes? DEFAULT: FALSE

Details

defaults to practice useR lightning talks: 15 slides, each shown 20 secs, change automatically

Value

none

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, June 2015

References

http://user2015.math.aau.dk/lightning_talks

See Also

alarm, Sys.sleep, txtProgressBar

Examples

```
timer(interval=0.5, n=3)
timer(interval=0.2, n=8, write=TRUE) # a slight deviation occurs for a large n
# timer() # to practice lightning talks at useR! conferences
```

---

toupper1 *capitalize words*

Description

capitalizes the first letter of character strings using toupper

Usage

```
toupper1(x)
```

Arguments

x	Character vector

Details

Basically just a one-liner using toupper

Value

character string vector

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jul 2016

See Also

toupper, substr

Examples

```
toupper1("berry")
toupper1(c("berRy","likes to code"))
```

traceCall *call stack of a function*

Description

trace the call stack e.g. for error checking and format output for do.call levels

Usage

```
traceCall(skip = 0, prefix = "\nCall stack: ", suffix = "\n",
  vigremove = TRUE)
```

Arguments

skip	Number of levels to skip in traceback
prefix	Prefix prepended to the output character string. DEFAULT: "\nCall stack: "
suffix	Suffix appended to the end of the output. DEFAULT: "\n"
vigremove	Logical: remove call created using devtools::build_vignettes()? DEFAULT: TRUE

Value

Character string with the call stack

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sep 2016 + March 2017

See Also

tryStack, checkFile for example usage

Examples

```
lower <- function(a, s) {warning(traceCall(s), "stupid berry warning: ", a+10); a}
upper <- function(b, skip=0) lower(b+5, skip)
upper(3)
upper(3, skip=1) # traceCall skips last level (warning)
upper(3, skip=4) # now the stack is empty
d <- tryStack(upper("four"), silent=TRUE)
inherits(d, "try-error")
cat(d)

lower <- function(a,...) {warning(traceCall(1,"",": "), "stupid berry warning: ",
                                  a+10, call.=FALSE); a}
upper(3)
```

truncMessage *truncate message parts*

Description

truncate long vectors for messages

Usage

```
truncMessage(x, ntrunc = 3, prefix = "s", midfix = " ", altnix = "'")
```

Arguments

x	Character vector
ntrunc	Integer: number of elements printed before truncation. DEFAULT: 3
prefix	Character: Prefix added if length(x)>1. DEFAULT: "s"
midfix	Character: string added after prefix OR before first altnix. DEFAULT: " "
altnix	Character: Alternative string padded around x if length(x)==1. DEFAULT: "'"

Value

Character string

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Nov 2016

See Also

message

Examples

```
truncMessage("hi")
message("listing name", truncMessage(       "hi"      ), ".")
message("listing name", truncMessage(paste0("hi",1:10)), ".")
truncMessage(paste0("hi",1:10), ntrunc=1)
truncMessage(paste0("hi",1:10), ntrunc=2, prefix="", midfix="")
truncMessage(paste0("hi",1:10), ntrunc=8, prefix="files _ ")
```

tryStack *try an expression, returning the error stack*

Description

As in try, the result of an expression if it works. If it fails, execution is not halted, but an invisible try-error class object is returned and (unless silent=TRUE) a message catted to the console.
Unlike try, tryStack also returns the calling stack to trace errors and warnings and ease debugging.

Usage

```
tryStack(expr, silent = FALSE, warn = TRUE, short = TRUE, file = "",
  removetry = TRUE)
```

Arguments

expr	Expresssion to try, potentially wrapped in curly braces if spanning several commands.
silent	Logical: Should printing of error message + stack be suppressed? Does not affect warnings and messages. DEFAULT: FALSE
warn	Logical: trace warnings and messages also? They are still handled like regular warnings / messages unless file !="", when they are catted into that file. DEFAULT: TRUE
short	Logical: should trace be abbreviated to upper -> middle -> lower? If NA, it is set to TRUE for warnings and messages, FALSE for errors. DEFAULT: TRUE
file	File name passed to cat. If given, Errors will be appended to the file after two empty lines. if warn=T and file!="", warnings and messages will not be shown, but also appended to the file. This is useful in lapply simulation runs. DEFAULT: "" (catted to the console)
removetry	Logical: should all stack entries matching typical tryCatch expressions be removed? Unless the call contains customized tryCatch code, this can be left to the DEFAULT: TRUE

Value

Value of expr if evaluated successfully. If not, an invisible object of class "try-error" as in try with the stack in object[2]. For nested tryStack calls, object[3], object[4] etc. will contain "– empty error stack –"

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Nov 2016

See Also

try, traceCall, http://r.789695.n4.nabble.com/Stack-trace-td4021537.html, http://stackoverflow.com/questions/15282471/get-stack-trace-on-trycatched-error-in-r, http://stackoverflow.com/questions/1975110/printing-stack-trace-and-continuing-after-error-occurs-in-r, http://stackoverflow.com/questions/16879821/save-traceback-on-error-using-trycatch

Examples

```
# Functions -----

lower <- function(a) {message("fake message, a = ", a); a+10}
middle <- function(b) {plot(b, main=b) ; warning("fake warning, b = ", b); lower(b) }
upper <- function(c) {cat("printing c:", c, "\n") ; middle(c)}
d <- upper(42)
d
rm(d)


# Classical error management with try -----

## Not run:  ## intentional error
d <- upper("42")               # error, no d creation
traceback()                    # calling stack, but only in interactive mode

## End(Not run)

d <- try(upper("42"), silent=TRUE)      # d created
cat(d)                                  # with error message, but no traceback
inherits(d, "try-error")                # use for coding


# way cooler with tryStack -----

d <- tryStack(upper("42"), silent=TRUE) # like try, but with traceback, even for warnings
cat(d)
d <- tryStack(upper("42"), silent=TRUE, warn=0) # don't touch warnings

tryStack(upper(42)) # returns normal output, but warnings are easier to debug
# Note: you can also set options(showWarnCalls=TRUE)

stopifnot(inherits(d, "try-error"))
stopifnot(tryStack(upper(42))==52)

## Not run:  ## file writing not wanted by CRAN checks
d <- tryStack(upper("42"), silent=TRUE, file="log.txt")
system2("open", "log.txt")
unlink("log.txt")

## End(Not run)
```

```
# Nested calls -----

f <- function(k) tryStack(upper(k), silent=TRUE)
d <- f(42)                ; cat("-----\n", d, "\n-----\n") ; rm(d)
d <- f("42")              ; cat("-----\n", d, "\n-----\n") ; rm(d)
d <- tryStack(f(4) )      ; cat("-----\n", d, "\n-----\n") ; rm(d)
# warnings in nested calls are printed twice, unless warn=0
d <- tryStack(f(4), warn=0) # could also be set within 'f'

d <- tryStack(f("4"))     ; cat("-----\n", d, "\n-----\n")
d[1:3] ; rm(d)
# empty stack at begin - because of tryStack in f, no real error happened in f


# Other tests -----

cat( tryStack(upper("42")) )
f <- function(k) tryStack(stop("oh oh"))
d <- f(42) ; cat("-----\n", d, "\n-----\n") ; rm(d) # level 4 not helpful, but OK

# stuff with base::try
f <- function(k) try(upper(k), silent=TRUE)
d <- f(42)    ; cat("-----\n", d, "\n-----\n") ; rm(d)
d <- f("42")  ; cat("-----\n", d, "\n-----\n") ; rm(d) # regular try output

f2 <- function(k) tryStack(f(k), warn=0, silent=TRUE)
d <- f2(42)   ; cat("-----\n", d, "\n-----\n") ; rm(d)
d <- f2("42") ; cat("-----\n", d, "\n-----\n") ; rm(d) # try -> no error.
# -> Use tryCatch and you can nest those calls. note that d gets longer.
```

`unitHydrograph` *Unit Hydrograph*

Description

Calculate continuous unit hydrograph with given n and k (in the framework of the linear storage cascade)

Usage

```
unitHydrograph(n, k, t, force = FALSE)
```

Arguments

n	Numeric. Number of storages in cascade.
k	Numeric. Storage coefficient [1/s] (resistance to let water run out). High damping = slowly reacting landscape = high soil water absorbtion = high k.

t	Numeric, possibly a vector. Time [s].
force	Logical: Force the integral of the hydrograph to be 1? DEFAULT: FALSE

Value

Vector with the unit hydrograph along t

Note

The sum under the UH should always be 1 (if t is long enough). This needs yet to be checked...

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, July 2013

See Also

lsc on how to estimate n and k for a given discharge dataset. deconvolution.uh in the package hydromad, http://hydromad.catchment.org

Examples

```
Time <- 0:100
plot(Time, unitHydrograph(n=2,  k=3, t=Time), type="l", las=1,
     main="Unit Hydrograph - linear storage cascade")
lines(Time, unitHydrograph(n=2,  k=8, t=Time), col=2)
lines(Time, unitHydrograph(n=5.5,k=8, t=Time), col=4)
text(c(12, 20, 50), c(0.1, 0.04, 0.025), c("n=2, k=3","n=2, k=8","n=5.5, k=8"),
     col=c(1,2,4), adj=0)

# try several parameters (e.g. in Monte Carlo Simulation to estimate
  # sensitivity of model towards slight differences/uncertainty in parameters):
nreps <- 1e3 # 5e4 eg on faster computers
n <- rnorm(nreps, mean=2, sd=0.8); n <- n[n>0]
k <- rnorm(nreps, mean=8, sd=1.1); k <- k[k>0]
UH <- sapply(1:nreps, function(i) unitHydrograph(n=n[i], k=k[i], t=Time))
UHquant <- apply(UH, 1, quantile, probs=0:10/10, na.rm=TRUE)
if(interactive()) View(UHquant)

plot(Time, unitHydrograph(n=2, k=8, t=Time), type="l", ylim=c(0, 0.06), las=1)
# uncertainty intervals as semi-transparent bands:
for(i in 1:5)
   polygon(x=c(Time, rev(Time)), y=c(UHquant[i,], rev(UHquant[12-i,])),
           col=rgb(0,0,1, alpha=0.3), lty=0)

lines(Time, UHquant[6,], col=4)
lines(Time, unitHydrograph(n=2, k=8, t=Time))

# Label a few bands for clarity:
points(rep(24,3), UHquant[c(2,5,9),25], pch="+")
for(i in 1:3) text(25, UHquant[c(2,5,9)[i],25],
```

```
                        paste("Q", c(10,40,80)[i], sep=""), adj=-0.1, cex=0.7)
# And explain what they mean:
Explain <- "Q80: 80% of the 50000 simulations are smaller than this value"
legend("topright", bty="n", legend=Explain)

# Some n and k values are cut off at the left, that explains the shift from the
# median of simulations relative to the n2k8 line.
```

`yearSample` *Nonrandom year with sample*

Description

Nerdy way to wish someone a happy new year by using sample

Usage

```
yearSample(year)
```

Arguments

`year`	4 digit numerical year.

Details

Nerdy way to wish someone a happy new year, eg:
Have a great
`set.seed(1244); sample(0:9,4,T)`

Value

`cat`s command into the console that can be copypasted to anyone's R script.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, April 2014

See Also

`nameSample` to impress with "randomly" finding a name, `set.seed`, `sample`, `letters`

Examples

```
yearSample(2016)
# Have a nerdy
set.seed(12353); sample(0:9, 4, replace=TRUE)
```

Index