Package ‘berryFunctions’

May 1, 2019

Type Package
Title Function Collection Related to Plotting and Hydrology
Version 1.18.2
Date 2019-04-29
Imports grDevices, graphics, stats, utils, abind
Suggests RColorBrewer, pbapply, knitr, rmarkdown, gstat, RCurl, colorspace
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/berry/berryFunctions
RoxygenNote 6.1.1
VignetteBuilder knitr
BugReports https://github.com/berry/berryFunctions
NeedsCompilation no
Repository CRAN
Date/Publication 2019-05-01 19:40:07 UTC

R topics documented:

berryFunctions-package .................................................. 4
addAlpha ................................................................. 5
addFade ................................................................. 6
addRows ................................................................. 7
almost.equal ........................................................... 8
anhang ................................................................. 9
R topics documented:

approx2 .................................................. 10
around ..................................................... 11
betaPlot ................................................... 12
betaPlotComp ............................................. 13
between .................................................... 15
catPal ..................................................... 16
checkFile ................................................ 17
ciBand ..................................................... 18
circle ...................................................... 20
classify ................................................... 21
climateGraph ............................................ 23
colPoints ............................................... 26
colPointsHist .......................................... 30
colPointsLegend ....................................... 32
combineFiles ........................................... 34
compareFiles ........................................... 35
convertUmlaut ......................................... 36
createFun ............................................... 37
createPres .............................................. 38
dataStr .................................................. 39
distance .................................................. 40
divPal ..................................................... 41
dupes ..................................................... 42
exp4p ...................................................... 43
expReg .................................................... 45
exTime ................................................... 46
funnelPlot ................................................ 48
funSource ............................................... 51
getColumn ............................................... 52
getName .................................................. 54
gof ......................................................... 55
googleLink2pdf .......................................... 57
groupHist ............................................... 59
headtail .................................................. 60
horizHist ................................................ 61
insertRows .............................................. 63
instGit .................................................... 64
is.error ................................................... 65
l2array ................................................... 66
l2df ....................................................... 68
learnVocab .............................................. 71
legendmt ................................................ 72
library2 .................................................. 73
lim0 ....................................................... 74
linLogHist ............................................... 75
linLogTrans ............................................ 77
linReg ..................................................... 79
locArrow .................................................. 81
### R topics documented:

- locLine ................................................................. 82
- logAxis ................................................................. 83
- logHist ................................................................. 85
- logSpaced ............................................................. 87
- logVals ................................................................. 88
- lsc ......................................................................... 90
- lsMem ................................................................. 93
- monthAxis ............................................................ 94
- monthLabs ........................................................... 96
- movAv ................................................................. 97
- movAvLines .......................................................... 99
- mReg ................................................................. 100
- na9 ............................................................... 106
- nameSample ......................................................... 107
- newFilename ........................................................ 108
- normalizePathCP .................................................... 109
- normPlot ............................................................. 110
- openFile ............................................................. 112
- owa ................................................................. 113
- packagePath ........................................................ 114
- panelDim ............................................................. 115
- parallelCode ......................................................... 117
- pdfpng ............................................................... 117
- popleaf .............................................................. 119
- pretty2 .............................................................. 120
- quantileBands ....................................................... 121
- quantileMean ......................................................... 123
- rainbow2 ............................................................. 125
- removeSpace ......................................................... 126
- rescale ............................................................... 127
- round0 ............................................................... 128
- roundedRect ........................................................ 129
- runAxis .............................................................. 130
- seasonality .......................................................... 131
- seqPal ............................................................... 135
- seqR ................................................................. 136
- showPal .............................................................. 137
- smallPlot ............................................................ 138
- smoothLines ......................................................... 140
- sortDF ............................................................... 142
- spiralDate ........................................................... 143
- spiralDateAnim ...................................................... 144
- superPos ............................................................. 146
- tableColVal .......................................................... 147
- testExamples ........................................................ 149
- textField ............................................................ 150
- TFtest ............................................................... 153
- timeAxis ............................................................. 154
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

dataDWD and readDWD have moved to the package rdwd: https://github.com/brry/rdwd

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 non-executable 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!

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2011-2017
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)
    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-z
    plot(x, type="n")
addFade

Description
Make existing colors fade away to white

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td>Vector of color names (colors), hexadecimal or integer that can be interpreted by col2rgb</td>
</tr>
<tr>
<td>fade</td>
<td>Level of fading towards target. between 0 (target) and 1 (col). Can also be a vector. DEFAULT: 0.3</td>
</tr>
<tr>
<td>target</td>
<td>Target color that should be faded into. DEFAULT: &quot;white&quot;</td>
</tr>
<tr>
<td>...</td>
<td>Further arguments passed to colorRamp</td>
</tr>
</tbody>
</table>

Value
character matrix with hexadecimal color codes.

Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Feb 2016

See Also
addAlpha, colorRamp, colors

Examples
```r
plot(1:11, pch=16, cex=3, col=addFade(2, 10:0/10))
plot(1:11, pch=16, cex=3, col=addFade(2, 10:8/10, target="blue"))
plot(1:11, pch=16, cex=3, col=addFade(2, 10:8/10, 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, sortDF, 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)                                    # integer numeric see above
0.4+0.4i - 0.1-0.1i == 0.3+0.3i  # complex
all.equal(0.4+0.4i - 0.1-0.1i, 0.3+0.3i)               # complex
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)                                         # 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)                                         # list
almost.equal(A,B)                                      # list

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

approx2(c(NA,NA))  # yields a message
approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1)) # fills with first non-NA value
approx2(c( 2,NA, 6, 4, 8, 9, 3, 2, 1)) # interpolates linearly
approx2(c( 2, 4, 6, 4, 8, 9,NA, 2,NA)) # linear, then last non-NA at end

approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1))
approx2(c(NA,NA, 6, 4, 8, 9, 3, 2, 1), fill=median) # first median, then linear
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)), 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(0.3, 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**

```r
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
```

---

**Description**

Visually understand the effect of the beta distribution parameters

**Usage**

```r
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**

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

Are values between a and b?

Description

Are values within a certain interval? Basically a wrapper for \( x \geq a \ \& \ x \leq b \) to save repeating long x names twice.

Usage

between(x, a, b = a, incl = TRUE, aincl = incl, bincl = incl, quiet = FALSE)

Arguments

x          Numerical vector
a, b       Numerical values/vectors specifying the borders of the interval. min and max are used, so they can be a vector.
incl       Logical. Include values on the borders? For x == border, TRUE will be returned. Specify per left and right border separately with the arguments aincl and bincl. DEFAULT: TRUE
aincl, bincl Logical. Include values on left and right border, respectively? DEFAULT: incl
quiet       Logical. Suppress warning if a>b? DEFAULT: FALSE

Value

Logical (boolean) vector with TRUE/FALSE values

Author(s)

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

See Also

findInterval

Examples

between(1:10, 4, 8)
between(1:10, 4:8) # range as vector
between(1:10, 8, 4) # warns about interval

data.frame( incl.T=between(1:10, 4, 8),
            incl.F=between(1:10, 4, 8, incl=FALSE),
            aincl.F=between(1:10, 4, 8, aincl=FALSE),
            bincl.F=between(1:10, 4, 8, bincl=FALSE) )
catPal

*Categorical color palette*

---

**Description**


**Usage**

```r
catPal(n = 12, set = 1, alpha = 1)
```

**Arguments**

- `n` Number of colors, max 12. DEFAULT: 12
- `set` Integer for which set to use. Currently, only 1 is implemented.
- `alpha` Transparency (0=transparent, 1=fully colored). DEFAULT: 1

**Value**

Character string vector with color names

**Author(s)**

Berry Boessenkool, <berry-b@gmx.de>, Apr 2019

**See Also**

`showPal`, `seqPal`, `divPal`

**Examples**

```r
plot(rep(1,12), pch=16, cex=5, col=catPal(12), xaxt="n")
showPal()
plot(cumsum(rnorm(40)), type="l", col=catPal()[1], ylim=c(-10,10))
for(i in 2:6) lines(cumsum(rnorm(40)), col=catPal()[i])
```
Description

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

Usage

checkBox(file, warnonly = FALSE, trace = TRUE, pwd = 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
pwd Logical: Print working directory in message? DEFAULT: TRUE

Value

TRUE/FALSE, invisibly

Author(s)

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

See Also

file.exists

Examples

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

checkBox("./", warnonly=TRUE)
checkBox(c("./","./"), warnonly=TRUE)

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

is.error(compareFiles("dummy.nonexist", "dummy2.nonexist"), TRUE, TRUE)
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,
        lwd = 1, 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 middle/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

is.error(checkFile("dummy.nonexist"), TRUE, TRUE)

## 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)
ciBand

add
Add to existing plot? If FALSE, plot is called before adding confidence interval. DEFAULT: FALSE

lwd
Line width of middle line. DEFAULT: 1

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

```r
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(y1=y1, yu=y2, ym=y3)

y1[6:7] <- NA
ciBand(y1=y1, yu=y2, ym=y3) # interpolation marked with stars if nastars=TRUE
ciBand(y1=y1, yu=y2, ym=y3, na="remove")
lines(y1, col=3, type="o")
lines(y2, col=3, type="o")

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

# Actual usefull stuff: sample size dependency of max and mean
```
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
classify

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 = "linear", breaks = NULL, Range = range(x, finite = TRUE), sdlab = 1, logbase = 1, quiet = FALSE)

Arguments

x  Vector with numeric values
method  Character string (partial matching is performed). Classification method (type of binning) to compute the class breakpoints. See section Details. DEFAULT: "linear"
breaks  Specification for method, see Details. DEFAULT: NULL (different defaults for each method)
Range  Ends of intervals. 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. DEFAULT: 1
logbase  base for logSpaced. Used only if not 1 and method="log". DEFAULT: 1
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).

<table>
<thead>
<tr>
<th>method</th>
<th>explanation</th>
<th>meaning of breaks</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td><em>nbins</em> equally spaced classes</td>
<td><em>nbins</em></td>
<td>100</td>
</tr>
<tr>
<td>log</td>
<td><em>nbins</em> logarithmically spaced</td>
<td><em>nbins</em></td>
<td>100</td>
</tr>
<tr>
<td>quantile</td>
<td>classes have equal number of values</td>
<td>the quantiles (or number of them)</td>
<td>10:4/4</td>
</tr>
<tr>
<td>sd</td>
<td>normal distributions</td>
<td>number of sd in one direction from the mean</td>
<td>3</td>
</tr>
<tr>
<td>custom</td>
<td>user-given breakpoints</td>
<td>breakpoint values (including ends of Range)</td>
<td>none</td>
</tr>
</tbody>
</table>

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:

See Also

*colPoints*

Examples

```r
classify( c(1:10, 20), "lin", breaks=12)
classify( c(1:10, 20), "q", breaks=0:10/10)
classify( c(1:10, 20), "s", sdb=2 )
classify( c(1:10, 20), "s", sdb=1, breaks=2 )
classify( c(1:10, 20), "c", breaks=c(5,27) )
classify( c(1:10, 20), "log")

set.seed(42); rz <- rnorm(30, mean=350, sd=120)
plot(1)
classileg <- function(method="linear", breaks=100, sdb=1, logbase=1, ...)```
### Description

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

### Usage

```r
climateGraph(temp, rain, 
  main = "StatName\n52°U\008058' E\n42 m aSL", 
  units = c("°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), keepar = 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\n52°U\008058' E\n42 m aSL"
- **units**: units used for labeling. DEFAULT: c("d C", "mm")
- **labs**: labels for x axis. DEFAULT: J,F,M,A,M,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 labeling? (not recommended for casual visualization!). DEFAULT: FALSE
- **ticklab**: positions for vertical labeling. 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 background 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:
https://www.hoelzel.at/__verlag/geojournal/archiv/klima/2006_01/lieth.gif
https://www.hoelzel.at/__verlag/geojournal/archiv/klima/istanbul/istanbul400.gif
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, argrid=list(col="gold"))
# for the Americans - axes should be different, though!: climateGraph(temp, rain, units=c("\U{00026}F","\U{000020}"))

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)) # multipanel 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 heights 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="green"))

# example with (fake) weekly relative soil moisture (RSM) added:
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)
set.seed(3)
soil <- berryFunctions::rescale(cumsum(rnorm(52)), from=1, to=100)
xsoil <- seq(1, 12, length.out=52)

climateGraph(temp, rain, ylim=c(-10, 50) ) # ylim for RSM 0:100 on second axis
lines(xsoil, soil/2, lwd=5, col="orange")

mtext(paste("Relative\nsoil moisture\n\U{0000D8}", round(mean(soil), 1), ", 
\U{0025B3}"),
side=3, col="orange", line=1, adj=0.99)

## Not run:

dev.off()

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

# Climate Graphs for the USA:
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, nrow=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=500, nrow=1, as.is=TRUE)[1,-1]
RP <- as.numeric(substr(RP,1,nchar(RP)-1))/100*25.4
meta <- read.table(file="Roseburg.txt", nrow=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)

cache(RT=as.numeric(RT), RP=as.numeric(RP), meta=meta)
# Climate Graphs for Germany:
browseURL("https://github.com/berry/rdwd#rdwd")
link <- rdwd::selectDWD("Potsdam", res="monthly", var="kl", per="h")
file <- rdwd::dataDWD(link, dir=tempdir(), read=FALSE)
clim <- rdwd::readDWD(file)
rdwd::readVars(file)
temp <- tapply(clim$MO_TT, INDEX=format(clim$MESS_DATUM, "%m"), FUN=mean, na.rm=FALSE)
precums <- tapply(clim$MO_RR, INDEX=format(clim$MESS_DATUM, "%Y-%m"), FUN=sum)
eachmonth <- format(strptime(paste(names(precums),"01"), "%Y-%m %d","%m")
prec <- tapply(precums, eachmonth, FUN=mean, na.rm=TRUE)
meta <- paste("Potsdam\n", paste(range(clim$MESS_DATUM, na.rm=TRUE), 
  collapse=" to "), "\n", sep="\n")

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$MO_TT, INDEX=format(clim$MESS_DATUM, "%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)

TQ <- sapply(TQ, I)
arrows(x0=1:12, y0=TQ["25%"], y1=TQ["75%"], 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
# library2("climatol") # commented out to avoid dah error in dataStr testing
# data(datcli)
# diagwl(datcli,est="Example station",alt=100,per="1961-90",mlab="en")

## End(Not run)
colPoints

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.
Adding points after smallPlot is called for the legend may be incorrect if the original function
messes with the graph margins, see the note in colPointsLegend.

Usage

colPoints(x, y, z, data, add = TRUE, col = seqPal(100), col2 = c(NA,
"grey", "black"), Range = range(z, finite = TRUE), method = "linear",
breaks = length(col), sdlab = 1, legend = TRUE, legargs = NULL,
lines = FALSE, nint = 30, xlab = gsub("\"","",
deparse(substitute(x))), ylab = gsub("\"","",
deparse(substitute(y))), zlab = gsub("\"","",
deparse(substitute(z))), axes = TRUE, log = "", las = 1,
bglines = NULL, pch = 16, x1 = 0.6, y1 = ifelse(horizontal, 0.88,
0.3), x2 = 0.99, y2 = 0.99, density = NULL, horizontal = TRUE,
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 colPoints, 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. DEFAULT:
"linear"
breaks Specification for method, see classify. DEFAULT: different defaults for each
method
sdlab Type of label and breakpoints if method="sd", 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
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 is 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, ylab, zlab
X axis label, y axis label, colPointsLegend title. DEFAULT: gsub("\n", ",", deparse(substitute(xlab)), gsub("\n", ",", deparse(substitute(ylab)), gsub("\n", ",", deparse(substitute(zlab)))))

axes, las
Draw axes? Label Axis Style. Only used when add=FALSE. See par. DEFAULT: axes=TRUE, las=1 (all labels horizontal)

log
Logarithmic axes with log="y", "xy" or "x". DEFAULT: ""

bglines
If not NULL, passed to abline to draw background lines before adding colored points. DEFAULT: NULL

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, or FALSE to suppress drawing it. DEFAULT: NULL

horizontal
Logical passed to colPointsLegend. DEFAULT: TRUE

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

See Also
classify, colPointsLegend, colPointsHist
Examples

\[
i <- c(22, 40, 48, 68, 80, 70, 70, 63, 55, 48, 45, 40, 30, 32)
j <- c(5, 10, 15, 20, 12, 30, 45, 40, 30, 36, 56, 33, 45, 23)
k <- c(175, 168, 163, 132, 120, 117, 110, 130, 131, 160, 105, 174, 190, 183)
\]

# 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 colR

# 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

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

# add single newly measured points to image (fictional data):
mx <- c(22, 40, 45, 30, 30, 10)
my <- c(5, 33, 56, 70, 45, 45)
mz <- c(110, 184, 127, 133, 170, 114)
colPoints(mx,my,mz, cex=5, pch="*", Range=c(94, 195), col=seqPal(), col2=NA, legend=FALSE)
points(mx,my, cex=4)
text(mx,my,mz, adj=-0.5, font=2)

# 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, lwd=3)
colPoints(x,y,n, data=rivers, add=FALSE, lines=TRUE, pch=3, lwd=3, nint=2)
colPoints("x","y","n", data=rivers, add=FALSE)

# different classification methods:
# see ?classify

colPoints(i,j,k, add=FALSE) # use classify separately:
### colPointsHist

*Histogram for colPoints*

**Description**

Adds Histogram to plots created or enhanced with `colPoints`

```r
# Add histogram:
cp <- colPoints(i,j,k, add=FALSE)
do.call(colPointsHist, cp[c("z","at","labels","bb","nbins")])
do.call(colPointsHist, owa(cp[c("z","at","labels","bb","nbins")],
                 list(bg=5, breaks=5)))
do.call(colPointsHist, owa(cp[c("z","at","labels","bb","nbins")],
                 list(mar=c(0,0,0,0), x1=0.5, x2=1, y1=0.8,
                   y2=0.99, yaxt="n")))

# 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=divPal(100), cex=1.2)))

# 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, add=FALSE)
}
```
Usage

```r
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
colPointsLegend

Description

Adds legends to plots created or enhanced with \texttt{colPoints}.
\texttt{sf} plots set \texttt{par(mar=c(0,0,1.2,0))} but then reset it to the values before. \texttt{smallPlot} will hence also reset to that, so points added after calling \texttt{colpointsLegend} will be wrong, unless the margins are set BEFORE \texttt{sf plot}. \texttt{sf:::plot.sf} alternatively uses \texttt{c(2.1, 2.1, 1.2, 0)} or \texttt{c(1, 1, 1.2, 1)}.

Usage

\begin{verbatim}
colPointsLegend(z, Range = range(z, finite = TRUE), nbins = 100,
               colors = seqPal(nbins), bb = seqR(Range, length.out = nbins + 1),
               nlab = 5, at = pretty2(Range, nlab), 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 = par("bg"),
               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, above, below, ...)
\end{verbatim}

Arguments

\begin{enumerate}
\item \texttt{z} \hspace{1cm} Values of third dimension used in \texttt{colPoints}, can be a matrix or a vector etc, but must be numeric
\item \texttt{Range} \hspace{1cm} Ends of color bar for method=equalinterval. DEFAULT: range(z, finite=TRUE)
\item \texttt{nbins} \hspace{1cm} Number of classes (thus, colors). If \texttt{colors} is given, \texttt{nbins} is overwritten with length(colors). DEFAULT: 100
\item \texttt{colors} \hspace{1cm} Color vector. DEFAULT: \texttt{seqPal} from yellow (lowest) to blue (highest value in \texttt{Range})
\end{enumerate}
**colPointsLegend**

This function adds a legend or key to a plot. It can be used with or without an inset plot, and it can be placed relative to the current figure or the device.

- **bb**: Borders of bins for the legend (key). DEFAULT: `seqR(Range, length.out=nbins+1)`
- **nlab, at, labels**: Number of legend labels, their positions and labels. DEFAULT: `nlab=5, labels=at=pretty2(Range,nlab)`
- **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 outside 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: `par("bg")`
- **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, above, below**: 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(), outFile = "combined_Textfiles.txt",
overwrite = FALSE, sep = NULL, names = TRUE, selection = NULL,
progbar = !quiet, quiet = FALSE, ...)

Arguments

inFiles vector with names of input files, as can be read with scan. DEFAULT: dir()
outFile Character string: name of the file to be created. Passed to newFilename. DE-
FAULT: "combined_Textfiles.txt"
overwrite Logical: overwrite outFile? DEFAULT: FALSE
compareFiles

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

...  Arguments passed to scan, but not one of: file, what, blank.lines.skip, sep, quiet.

Value

Final output file, invisibly.

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(infile=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

\begin{verbatim}
compareFiles(file1, file2, nr = 20, startline = 1, endline = length(f1), quiet = FALSE, ...)
\end{verbatim}

Arguments

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

Value

Vector of line numbers that differ, result from \texttt{head}(..., nr)

Author(s)

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

See Also

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

Examples

\begin{verbatim}
filenames <- system.file(paste0("extdata/versuch",1:2,".txt"), package="berryFunctions")
compareFiles(filenames[1], filenames[2], warn=FALSE)
\end{verbatim}

\begin{verbatim}
convertUmlaut(x)
\end{verbatim}

Description

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

Usage

\begin{verbatim}
convertUmlaut(x)
\end{verbatim}

Arguments

- \texttt{x} Character string(s) containing German Umlaute
createFun

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

```r
## 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),])
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 a package

Usage

```r
createFun(fun, path = ".", open = TRUE)
```

Arguments

- **fun**: Character string or unquoted name. Function that will be created with identical filename.
- **path**: Path to package in development (including package name itself). Is passed to packagePath. DEFAULT: "."
- **open**: Logical: open the file? If several instances of Rstudio are open, the last one (not necessarily the active one) will be used. DEFAULT: TRUE

Details

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

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 figExtern folder.

Usage

createPres(presname = "pres", dir = "presentation", path = ".", navbullets = FALSE, bgblack = FALSE, open = TRUE)

Arguments

presname Name of .Rnw file to be created. DEFAULT: "pres"

dir Name of directory that will contain .Rnw file and figExtern folder. "_1" will be appended if already existing, see newFilename. DEFAULT: "presentation"

path Location of dir. Passed to setwd. DEFAULT: "."

navbullets Logical: include navigation slide bullet points in header? DEFAULT: FALSE

bgblack Logical: set a black background instead of a white one? Requires all R graphics fg and bg colors to be changed! See "How to avoid death By PowerPoint" at 11:49 minutes https://youtu.be/IwpillM6dF0?t=11m49s. Change colors manually in the Rnw files searching for bg=, linkcolor=, urlcolor= in the preamble and color right after begin document. DEFAULT bgblack: FALSE

open Logical: run openFile? DEFAULT: TRUE

Author(s)

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

See Also

createFun

Examples

## Not run:
createPres("Berry_Conference")

## End(Not run)

dataStr str of datasets

Description

Print the str of each dataset returned by data

Usage

dataStr(package = NULL, df = FALSE, ...)

Arguments

package Package name. DEFAULT: NULL

df Logical: give information only about all data.frame objects? DEFAULT: FALSE

... other arguments passed to data

Value

invisible data.frame. Mainly 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

```r
# dataStr() # all loaded packages on search path (package=NULL)
dataStr("datasets") # only datasets in base R
dataStr("colorsname") # works with an installed but unloaded package

# data.frames only
d <- dataStr(df=TRUE)
head(d)
d[,c("Call","ncol","nrow")]
```

---

**distance**

*Distance between points*

### Description

Calculate distance between points on planar surface

### Usage

```r
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
**divPal**

**Examples**

```r
A <- c(3, 9, -1)
B <- c(7, -2, 4)
plot(A, B)
text(A, B, paste("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**

```r
divPal(n = 100, reverse = FALSE, alpha = 1, rwb = FALSE, ryb = FALSE, gp = FALSE, br = FALSE, colors = NULL, ...)
```

**Arguments**

- `n` Number of colors. DEFAULT: 100
- `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
- `gp` Use green-purple instead of the default. DEFAULT: FALSE
- `br` Use blue-red instead of the default. 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

The default palette is originally in 12 shades in the IPCC Assessment Report 5 Chapter 12 Fig 12.22, http://www.ipcc.ch/report/ar5/wg1/.
The green-purple and blue-red palettes are from NYtimes (originally with 8 shades), https://nyti.ms/2mL0o4J

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>File name (character string)</td>
</tr>
<tr>
<td>ignore.empty</td>
<td>Should empty lines be ignored? DEFAULT: TRUE</td>
</tr>
<tr>
<td>ignore.space</td>
<td>Should leading/trailing whitespace be ignored? DEFAULT: TRUE</td>
</tr>
<tr>
<td>tofile</td>
<td>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)</td>
</tr>
<tr>
<td>n</td>
<td>Show only the first n values if tofile=FALSE. DEFAULT: length(d)</td>
</tr>
</tbody>
</table>

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.
exp4p

Note

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

Author(s)

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

See Also

compareFiles

Examples

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

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

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

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, ...)

exp4p

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 tries 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

```r
## Not run: # Skip time consuming checks on CRAN
# 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

## 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

```r
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, 0.5), 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 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: FALSE.
- **pos1**: `xy.coords`-acceptable position of the formula. DEFAULT: "top".
- **pos2**: For numerical coordinates, this is the y-position. DEFAULT: NULL, as in `legend`.
**exTime**

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**

```r
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)
```

**Description**

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

**Usage**

```r
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**

element, system.time

**Examples**

```r
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 confidence 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
a1
Arguments for legend (text.col, bty, border, y.intersp, etc.). DEFAULT: NULL
...

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.


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 apha Wilson method first yielded wrong upper limits in my translation (it needs 0:1 instead of %). Thus I added the wikipedia formula:
Which other methods should I include? (That’s not the hard part anymore)

Author(s)

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

References

https://www.perceptualedge.com/articles/visual_business_intelligence/variation_and_its_discontents.pdf
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

```
funnelPlot

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 successful than Mary in success 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(lyt=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 slightly different (supposedly better) limits for small n:
funnelPlot(X,N, method="classic", a3=list(title="Standard Method"))
funnelPlot(X,N, add=TRUE, method="wilson", a3=list(lyt=2, col="red"),
a2=list(lyt=2, col="blue"), a1=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(lyt=2, col="red"), a2=list(lyt=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), local = FALSE)

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)

local  
Open offline version of the code? Lacks comments and original formatting of source code. DEFAULT: FALSE

Value

links that are also opened with browseURL

Author(s)

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

See Also

https://github.com/berry/rskey#rskey to add this as a keyboard shortcut

Examples

## Not run:  ## browser windows should not be opened in CRAN checks
library("berryFunctions")
funSource(colPoints)
funSource("head")
funSource("require", local=TRUE) # usefull when offline

funSource("OSMscale::earthDist") # works even for non-installed CRAN packages

## End(Not run)

## Not run:  # developmental testing
is.error(funSource("earthDist"), TRUE, TRUE) # Error for unloaded package
getColumn

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

gColumn(x, df, trace = TRUE, convnum = TRUE, quiet = FALSE)

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
convnum Logical: Convert numerical input (even if character) to Column name for that number?
quiet Logical: suppress non-df warning? DEFAULT: FALSE

Value

Vector with values in the specified column

Author(s)

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

See Also

getColumn

Examples

```r
head(stackloss)
ggetColumn(Air.Flow, stackloss)
ggetColumn("Air.Flow", stackloss)
ggetColumn(2, stackloss)
ggetColumn("2", stackloss) # works too...

# useful warnings:
ggetColumn(1, stackloss[0,])
ggetColumn(1, data.frame(NA=rep(NA,10)))

# Code returning a character works as well:
ggetColumn(c("Air.Flow","Acid.Conc")[1], stackloss)

# Can be used in functions to get useful messages:
upper <- function(x, select) getColumn(x, stackloss[select,])
upper(Water.Temp)
upper(2)
upper(2, select=0)

checkerr <- function(x) invisible(is.error(x, force=TRUE, tell=TRUE))

# Pitfall lexical scoping: R only goes up until it finds things:
upper2 <- function(xx) {xx <- "Timmy!"; getColumn(xx, stackloss)} # breaks!
checkerr( upper2(Water.Temp) ) # Column "Timmy" does not exist
# If possible, use "colname" with quotation marks.
# This also avoids the CRAN check NOTE "no visible binding for global variable"
upper3 <- function(char=TRUE)
{
  Sepal.Length <- stackloss
  if(char) head(getColumn("Sepal.Length", iris), 10)
  else head(getColumn( Sepal.Length, iris), 10)
}
checkerr( upper3(char=FALSE) )
upper3(char=TRUE) # use string "Sepal.Length" and it works fine.

# The next examples all return informative errors:
checkerr( upper(Water) ) # partial matching not supported by design
checkerr( getColumn("dummy", stackloss)) # no NULL for nonexisting columns
checkerr( getColumn(2, stackloss[,0]) ) # error for empty dfs
checkerr( getColumn(Acid, stackloss) ) # no error-prone partial matching
checkerr( getColumn(2:3, stackloss) ) # cannot be a vector
checkerr( getColumn(c("Air.Flow","Acid.Conc"), stackloss) )

#getColumn("a", tibble::tibble(a=1:7, b=7:1)) # works but warns with tibbles

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

getColumn("1", df, convnum=FALSE) # now gives second column
# as said, don't name column 2 as "1" - that will confuse people

# More on scoping and code yielding a column selection:
uppl <- function(coln, datf) {getColumn(substitute(coln), datf)[1:5]}
uppl(Sepal.Length, iris)
uppl("Sepal.Length", iris)
upp2("Sepal.Length", iris)
vekt <- c("Sepal.Length","Dummy")
# uppl(vekt[1], iris) # won't work if called e.g. by testExamples()
uppl2(vekt[1], iris)

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])

upper2 <- function(data) upper(data)
upper2("K")
upper2(K)

# getName only works correctly if x is not an evaluated object:
lower2 <- function(inp, assign=FALSE) {if(assign) inp <- inp; getName(inp)}
lower2(pi) # "pi"
lower2(pi, TRUE) # "3.14159265358979"

---

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 minimizing 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 = "")
`gof`  

```r
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**


**Examples**

```r
# 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)
}
leggGOF(-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")
```
```r
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) )

## Not run: # time consuming Simulation

# sample size bias
x <- 1:1000
y <- x+rrnorm(1000)
rmse(x,y) # 0.983
ssize <- rep(5:1000, 3)
sgofs <- sapply(ssize, function(n) c(rsquare(x,y), rmse(x,y)))
plot(ssize, sgofs[2,]) # RMSE: no bias, symmetric convergence

if(require(pbsapply)) 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 + rrnorm(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)
```

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

Character string 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

"&cad=rja&uact=8&ved=0CDIQFjAA&url=http%3A%2F%2Fcran.r-project.org", 
"%2Fdoc%2Fmanuals%2FR-intro.pdf&ei=Ny14UfHeOIXCsw6pIC4CA", 
"&usg=AFQjCNGejDwp1lor4tog0ZmQE4vz7ycK9zBa&bvm=bv.45580626,d.Yms")
googleLink2pdf(Link)

"&cad=rja&uact=8&ved=0ahUKEwjllfClavRAhWaN1AKHCGBjEQFggMAA", 
"&url=http%3A%2F%2Fstackoverflow.com%2Fquestions%2Ftagged%2Fr", 
"&usg=AFQjCNHYyj6Hjs6Lvczn9wMWxE3sv1CdqlQ&bvm=bv.142059868,d.ZWm")
googleLink2pdf(Link)

"&cad=rja&uact=8&ved=0ahUKEwjllfClavRAhWaN1AKHCGBjEQFggpM&", 
"&url=http%3A%2F%2Fstackoverflow.com%2Fquestions%2Ftagged%2F%3Ftag", 
"&usg=AFQjCNGkPHGx05qwKLLW4vRXdMk201hmig&bvm=bv.142059868,d.ZWM")
googleLink2pdf(Link)
groupHist

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 df, 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

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 according 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
horizHist

Horizonal 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, ...)
horizHist

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
insertRows

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, rcurrent = FALSE)

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
rcurrent Logical: should r specify the current rows of df, after which new is to be appended? If FALSE (the default for backwards compatibility), the rownumbers of the output (instead of the input) are r. I.e. new is inserted at, not after the rownumber. DEFAULT: FALSE

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)

References

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

addRows, sortDF

Examples

```r
existingDF <- as.data.frame(matrix(1:20, nrow=5, ncol=4))
existingDF
insertRows(existingDF, 2) # default new=NA is recycled
insertRows(existingDF, 2, rcurrent=TRUE) # after current line, not at it
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) # use addRows for this:
  addRows(existingDF, n=1)
insertRows(existingDF, 9) # pads NA rows inbetween

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

# 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
```

### Description

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

### Usage

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

### Arguments

- **pk**
  Character string in the from of "user/package"

- **cleanup**
  Remove downloaded ziptile 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/berry/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 returning an error

<table>
<thead>
<tr>
<th>Logical: Should the error message be printed via message? DEFAULT: FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>force  Logical: Should an error be returned if the expression is not an error? DEFAULT: FALSE</td>
</tr>
</tbody>
</table>

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) ))

---

**12array**  

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

12array(x, ...)

Arguments

x  List with arrays/data.frames. 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

```
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)

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)

# 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)

# list of dataframes:
LDf <- list(IR1=iris[1:5,1:2], IR2=iris[11:15,1:2], IR3=iris[21:25,1:2])
l2array(LDF)

# General intro to arrays ----- 

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

# Selection:
A1[,1:n2]
A1[,1:2]
A1["row2",] # result rotated against expectation -> transpose with t(...)
A1[,] # aggregation:
apply(A1, MARGIN=1:2, FUN=sum) # keep first two dimensions
apply(A1, MARGIN=c(1,3), FUN=sum) # aggregate over my_y -> row: 6, 22, 38
A1["row",,]
# 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))
al1(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)
**List to data.frame**

**Description**

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

**Usage**

```r
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/8799990/converting-given-list-into-dataframe](http://stackoverflow.com/questions/8799990/converting-given-list-into-dataframe)

**See Also**

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

**Examples**

```r
eglist <- list(AA=c(6,9,2,6), BB=1:8, CC=c(-3,2) )
eglist
l2df(eclist)  # names are even kept
l2df(eclist, byrow=FALSE)
class( l2df(eclist, 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(eclist))  # 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!
colnames(eg6$DD) <- colnames(eg6$AA)
do.call(rbind, eg6)
# check some of the links above for more solutions...
Description

spaced learning e.g. for vocabulary. Uses interactive questions.
Note: this currently clears the console!
Based on https://ncase.me/remember by Nicky Case.
At the beginning, new vocab will be asked, skip with empty ENTER.

Usage

learnVocab(vocfile = "C:/Dropbox/Sonstiges/Vokabeln.csv", minhours = 3, nnew = 3)

Arguments

vocfile  File with vocabulary (or whatever you want to learn). The first two lines must contain the learning day and date, see examples. The third line must contain LEVEL:known:new, the last two being (short) names, e.g. languages (known will be displayed first).
minhours  Minumal number of hours since the last test. If the time difference is less, nothing happens. This enables putting learnVocab in Rprofile without being asked stuff all the time at every restart of R. DEFAULT: 3 hours
nnew  Number of new entries to be added interactively at the start. They can still be skipped by writing nothing and pressing the ENTER key. DEFAULT: 3

Value

Updated vocab list, invisibly.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Apr 2019

Examples

## Not run: # Excluded from checks, works only interactively!
# initiate empty vocab list:
vocfile <- tempfile("myvocab",fileext=".csv")
cat("learning_day 1\n2019-04-19 17:43:47\nLEVEL;DE;FR\n", file=vocfile)

learnVocab(vocfile) # asks new vocab, then tests and changes level as needed

## End(Not run)
**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 are 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**

```r
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")
```

```r
# 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, 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**

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

**Arguments**

- `name`: Name of the package(s). Can be quoted, 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**

```r
## 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":

linLogHist

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
List of arguments passed to `logAxis` in the final plot. DEFAULT: NULL

Plot on linear scale first? DEFAULT: TRUE

Plot on logarithmic scale at the end? DEFAULT: TRUE

Write transformation value in lower right corner? DEFAULT: TRUE

Supply vector with values for transformation \((1/t)\). Overrides 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

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

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

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

`linLogTrans`

```r
x <- rlnorm(700, m=3)
hist(x, col=4)
hist(log10(x), xaxt="n"); logAxis(); 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:
library(animation)
```
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). DE-
FAULT: 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), mgp=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
value_t  Supply vector with values for transformation \((1/t)\). Overrides 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

"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?id=1282690259&postcount=4
http://sfew.websitetoolbox.com/post/show_single_post?id=1282691799&postcount=5

See Also

logVals

Examples

set.seed(42); x <- 10*rnorm(100, 3); y <- runif(100)
linLogTrans(x,y, steps=15, sleep=0.05)
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-3,3e-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)
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)
Description

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

Usage

```r
linReg(x, y = NULL, data = NULL, add = FALSE, digits = 2, pch = 16, col = "black", colline = "red",
colband = addAlpha(colline), level = 0.95,
plotrange = par("usr")[1:2], 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 \text{\&} R^2=r \text{\&} \text{RMSE}=e", respectively. If a value is negative, the complete respective entry is left away. 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 points. DEFAULT: "black"
- **colline**: Color of the regression line, see `par`. DEFAULT: "red"
- **colband**: Color of the confidence region band. DEFAULT: addAlpha(col)
- **level**: Confidence level, see `predict.lm`. DEFAULT: 0.95
- **plotrange**: x range for which regression line and uncertainty band should be plotted. Is passed to `seqR` and can hence be a vector. DEFAULT: `par("usr")[1:2]`
- **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
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

```r
locArrow

# 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

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

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

linReg(a,b)
linReg(a,b, ylab="Hallo", pch=1, colline=3, main="Regression by Berry")
linReg(a, b, pos1=15, pos2=0) # position of topleft corner of legend
linReg(a, b, pos1=NA) # to suppress legend
linReg(a, b, plotrange=5:20) # only for plotting, all data points are used!
linReg(a,b, digits=c(2,3,2,-1) ) # Do not write RMSE into legend

# Formula specification:
linReg(b-a)
linReg(Fertility-Education, data=swiss, col="blue", colline="green")
# col is for points, colline + colband for regression line + conf.int.

# 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, colline=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)
```
locLine

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-mb@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

locArrow, 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 = NA, big.mark = ",", decimal.mark = ".",
scientific = FALSE, exponent = 5, expobase = FALSE,
allbase = 1:9, box = TRUE, ...)
logAxis

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 labeling. DEFAULT: NULL (internally, logVals$labs

allticks
Place all intermediate ticklines at the axis (without labeling). 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: NA -> 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: "\n"
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

expobase
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
logHist

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,
...)

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^c(1:4), ylim=10^c(4,1), type="o", log="y") # reverse axis:
plot(10^c(1:5), log="y"); logAxis(4, exponent=3) # different treshold
plot(10^c(1:5), log="y"); logAxis(4, expn=3, base=c(1,2,5), expobase=TRUE)
plot(-8:5); logAxis(4, allbase=c(1,2,5)) # In case you want to mislead...
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.
For exact logarithmic spacing, use \texttt{10^seq(from=\log_10(1), to=\log_10(100), len=100)}

Usage

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

Arguments

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

# For exactly spacing logarithmically, use
plot(10^seq(from=log10(1), to=log10(100), len=100), log="y")
browseURL("https://stackoverflow.com/a/29963530")

---

logVals Create log-axis values and labels

Description

Create nice values and labels to write at logarithmic axes

Usage

logVals(from = -7, to = 7, range, base = 1, big.mark = "",
decimal.mark = ".", scientific = FALSE, exponent = Inf,
exponentbase = 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). Use base=NA to switch between 1 and c(1,2,5)
depending on range. DEFAULT 1
- **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
- **exponentbase**: Should "n * " be appended before 10^exp if n=1? DEFAULT: FALSE
- **allbase**: Base for $all$ (for horizontal lines). DEFAULT: 1:9
- **...**: Ignored arguments
logVals

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


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(lv, 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()
logVals(scient=TRUE)
logVals(exponent=5)
logVals(big.mark=" ")
logVals(big=".", dec=",")
**Description**

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

**Usage**

```r
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](http://ponce.sdsu.edu/onlineuhcascade.php)
- Skript 'Abflusskonzentration' zur Vorlesungsreihe Abwasserentsorgung I von Prof. Krebs an der TU Dresden

**See Also**

- `unitHydrograph`, `superPos`, `nse`, `rmse`
- deconvolution.uh in the package hydromad, [http://hydromad.catchment.org](http://hydromad.catchment.org)

**Examples**

```r
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)

## Not run: ## Time consuming tests excluded from CRAN checks

## Apply this to the validation event
dummy <- lsc(valid$P, valid$Q, area=3.4, plotsim=FALSE, type="1")
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="1", area=3.4, fit=60:90) # overestimates first peak
# Area cannot be right - is supposedly 17 km^2.

## 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
**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**

```r
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**


**See Also**

`object.size`, `ls`

**Examples**

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

## End(Not run)
```
Description

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

Usage

```r
monthAxis(side = 1, time = NA, origin = "1970-01-01", mlabels = substr(month.abb, 1, 1), yformat = "%Y", nmonths = 3, nym_half = 3.5, nym_none = 5, mceX = 0.7, yceX = 1, mtc1 = par("tcl"), ytc1 = par("tcl") - 1.7, mline = -1, yline = 0.2, las = 1, lrange = NA, trunc = NA, mpg = c(3, 1, 0), mt = NULL, ml = NULL, yt = NULL, yl = NULL, quiet = FALSE, ...)
```

Arguments

- **side**: Which **axis** is to be labeled? DEFAULT: 1
- **time**: Logical indicating whether the axis is **POSIXct**, not **Date**. DEFAULT: NA, meaning axis value >1e5
- **origin**: Origin for **as.Date** and **as.POSIXct**. DEFAULT: "1970-01-01"
- **mlabels**: Labels for the months. DEFAULT: J,F,M,A,M,J,J,A,S,O,N,D
- **yformat**: Format of year labels, see details in **strptime**. Use `yformat=" "` (with space) to suppress year labeling. DEFAULT: "%Y"
- **nmonths**: Minimum number of months required before a year at the axis boundary is labeled. DEFAULT: 3
- **nym_half**: Number of years on axis above which only every second month is labeled. DEFAULT: 3.5
- **nym_none**: Number of years on axis above which the months are not labeled. DEFAULT: 5
- **mceX**: cex.axis (letter size) for month labels. DEFAULT: 0.7
- **yceX**: cex.axis (letter size) for year labels. DEFAULT: 1
- **mtc1**: Month tick length (negative text line height units). 0 to suppress ticks. DEFAULT: par("tcl") = -0.5
- **ytc1**: Year tick length (negative text line height units). 0 to suppress ticks. DEFAULT: par("tcl")-1.7 = -2.2
- **mline**: Line of month labels. DEFAULT: -1
- **yline**: Line of year labels. DEFAULT: 0.2
- **las**: LabelAxisStyle for orientation of labels. DEFAULT: 1 (upright)
- **lrange**: Label range (two **Date** values). DEFAULT: NA = internally computed from `par("usr")`
**monthAxis**

trunc Vector with two values: Number of days/seconds to truncate at the left and right end of a range. DEFAULT: NA

mgp MargIn Placement. Suggested not to change this, since _tcl and _line defaults are chosen for the DEFAULT: c(3,1,0)

mt, ml, yt, yl Lists with further arguments passed to *axis*, like lwd, col.ticks, lwd.ticks, hadj, lty, separately for month ticks, month labels, year ticks, year labels. DEFAULT: NULL

quiet Suppress warning about short time axis? DEFAULT: FALSE

... Arguments passed to *axis* for all 4 elements.

**Value**

List with locations of month and year labels and ticks, each a Date vector.

**Author(s)**

Berry Boessenkool, <berry-b@gmx.de>, Feb + Dec 2015, Oct 2017

**See Also**

*monthLabs* for the numbercrunching itself, *timeAxis* for shorter or longer time frames, *axis.Date* with defaults that are less nice.

**Examples**

```r
set.seed(007) # for reproducibility
timePlot <- function(nydays, start="2013-08-25", ...)  
  plot(as.Date(start)+sort(c(0, sample(1:nydays, 50))),  
      cumsum(rnorm(51)), type="l", xaxt="n", ann=FALSE, las=1, ...)

timePlot(1100)
monthAxis()
monthAxis(1, nmonths=6, col.axis="red") # 2013 not labeled anymore
monthAxis(side=3, nym_half=2) # if axis > 2 years, label only partially

timePlot(2e3)
monthAxis() # long time series (>nym_none) only have years labeled
ma <- monthAxis(side=3, font=2)
abline(v=ma$mtics, col=8)
abline(v=ma$ytics) # vertical lines in graph - now add lines/points

timePlot(900)
monthAxis(side=3, mtcl=0) # no tick lines between months
monthAxis(yces=1.4, ytc=2, lwd.ticks=2)
monthAxis(yline=1, col.axis=4, col=4)
monthAxis(mcex=1, col.axis="red", yformat=" ") # no years labeled
timePlot(900)
monthAxis(nmonths=1) # year labeled for short period as well
```
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
npy 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
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 certain 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

timeAxis for nice labeling, timeAxis for automatic determination of npm/npy, `as.Date`, paste.

Examples

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

# see monthAxis for automatic plot labeling
```

---

**movAv**

*Moving average*

**Description**

Weighted moving average (running mean) with overlapping windows.

**Usage**

```r
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

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

Examples

```r
# 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 intensitiy ------------------------------------------------------
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**

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

```r
mReg(x, y = NULL, data = NULL, Poly45 = FALSE, exp_4 = FALSE,  
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, ...)  
```

**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.
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

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 behaviour of a system adequately!
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 cubic 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 $a \cdot e^{b \cdot (x+c)} + d$

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](http://rclickhandbuch.wordpress.com/rpackages)

**See Also**

glm, lm, optim

**Examples**

```r
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=rainbow(2(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)
\texttt{mReg}(x,y, \texttt{legargs=list(cex=0.7, x="topright"}}, \texttt{main="dangers of extrapolation"})
\texttt{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:
\texttt{mReg}(c(x,11,13,15), c(y,2,2,2), \texttt{xf="myX"}, \texttt{yf="myY"}, \texttt{Poly45=TRUE, legendform="name"})
\texttt{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.
\texttt{mReg}(x,y, pcol=8, ncol=0) \# no return to console
\# only plot a subset: best n fits, minimum fit quality, or user selection
\texttt{mReg}(x,y, pcol=8, ncol=2, nbest=4)
\texttt{mReg}(x,y, pcol=8, ncol=2, R2min=0.7)
\texttt{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:
\texttt{plot}(x,y, xlim=c(0,40))
\texttt{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:
\texttt{mReg}(j,k, \texttt{legargs=list(x="bottomright", inset=.05, bty="o"}), legendform="name")
\# Legend forms
\texttt{mReg}(j,k, \texttt{legargs=list(x="bottomright"), legendform="name"})
\texttt{mReg}(j,k, \texttt{legargs=list(x="bottomright"), legendform="form"})
\texttt{mReg}(j,k, \texttt{legargs=list(x="bottomright"), legendform="nameform"})
\texttt{mReg}(j,k, \texttt{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
\texttt{tfile <- system.file("extdata/Temp.txt", package="berryFunctions")}
\texttt{temp <- read.table(tfile, header=TRUE, dec="",)}
\texttt{head(temp)}
\texttt{plot(temp)}
\texttt{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=49e^(-0.031*(x - 0)) + 25 correct, judged from the model:
# Temp=T0 - Te *exp(k*t) + Tn with T0=73.76, Tend=26.21, k=-0.031
# optmethod="Nelder-Mead" # y=52e^(-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) # Formulae 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)
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 \texttt{read.table}.

Usage

\begin{verbatim}
na9(nspace = 5, base = c(-9999, -999, -9.99, -9.999), sep = c(",", ",.",), digits = 0:4, more = NULL, ...)
\end{verbatim}

Arguments

- \texttt{nspace} number of spaces prepended. DEFAULT: 5
- \texttt{base} Numeric: basic na.string numbers
- \texttt{sep} Separator string (comma or decimal point or both). DEFAULT: c("","",".",)
- \texttt{digits} Number(s) of zeros to be appended. DEFAULT: 0:4
- \texttt{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

\texttt{paste}

Examples

\begin{verbatim}
na9()
n9(nspace=0, sep=",")
n9(nspace=0, sep=",.", more=c(NA,"-"))) 
\end{verbatim}
nameSample

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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Character string. long strings (&gt;5) will compute a VERY long time!</td>
</tr>
<tr>
<td>progress</td>
<td>Logical. Monitor progress by printing a dot every 10000 tries? DEFAULT: TRUE for long names (nchar(name)&gt;3).</td>
</tr>
<tr>
<td>estimatetime</td>
<td>Estimate computation time? DEFAULT: nc&gt;4</td>
</tr>
<tr>
<td>continue</td>
<td>Continue without asking? DEFAULT: FALSE</td>
</tr>
</tbody>
</table>

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

```r
## Not run in RCM 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
```
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, overwrite = FALSE, 
tellignore = TRUE, pre = "", mid = "\n", end = "", 
quiet = FALSE, ntrunc = 3)

Arguments

filename Char (vector): file name(s).
ignore Logical (vector, recycled): Ignore file? DEFAULT: FALSE
overwrite Logical (vector, recycled): overwrite file? DEFAULT: FALSE
tellignore Logical: Message about ignored files? DEFAULT: TRUE
pre, mid, end Char: strings to append after traceback / message / filenames. DEFAULT: ",", ",","
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

```r
fns <- c("dummy1", "dummy2.txt", "berryFunctions.Rproj", 
  "README.md", "dummy2.dummy", "DESCRIPTION", "dummy4.R", "dummy5")
newFilename(fns)
newFilename(fns, ignore=TRUE)
newFilename(fns, ignore=rep(c(TRUE,FALSE), each=4) )
newFilename(fns, ignore=rep(c(TRUE,FALSE), each=4), tellignore=FALSE)
newFilename(fns, ntrunc=2)
newFilename(fns, overwrite=TRUE, ign=c(TRUE,TRUE,rep(FALSE,6)))
newFilename("README.md")
newFilename("dummy", mid=" ") # no line break
```

normalizePathCP normalizePath Cross Platform

Description

`normalizePath` Cross Platform: Returns absolute path even for not (yet) existing files even on Linux. On Windows, this is the default behaviour.

Usage

`normalizePathCP(path, winslash = "/", mustWork = FALSE)`

Arguments

- `path` Character vector of file paths
- `winslash` Path separator on Windows. DEFAULT: "/" (unlike `normalizePath`)
- `mustWork` Logical for `normalizePath`. DEFAULT: FALSE

Value

`path` character string(s)
Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Nov 2017

See Also
normalizePath, newFilename

Examples

normalizePath("doesnotexist.file", mustWork=FALSE) # on unix not full path
normalizePathCP("doesnotexist.file") # full path on all platforms

normalizePath("./doesnotexist.file", mustWork=FALSE)
normalizePathCP("./doesnotexist.file")

checknp <- function(a,b=a,d=getwd())
{
  aa <- normalizePathCP(a)
  bb <- if(d="") b else paste0(d,"",b)
  if(aa != bb) stop("",a,"\' -> ",aa,"", should be ",bb,"-")
  aa
}

checknp("notexist.file")
checknp("./notexist.file", "notexist.file", dirname(getwd()))
checknp("notexistfolder/notexist.file")
#checknp("/home/berry/notexist.file", d="") # fails on windows
#checknp("S:/Dropbox/notexist.file",d="") # fails on linux

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 \nsd =", 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

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

open file in default application

Description
open a file using system2 with command based on operating system. Tries to open the file with the program associated with its file extension.

Usage
openFile(file, ...)

Arguments
file Filename to be opened, as character string.
...
Further arguments passed to system2

Value
Result of try(system2, ...), invisibly

Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Aug 2017

References

See Also
system2, checkFile

Examples
## Not run: # excluded from CRAN checks, file opening not wanted
openFile("README.md")
openFile("Tests.R")
is.error(openFile("dummydummydoesntexist.R"), TRUE, TRUE)
openFile(tempdir())

## End(Not run)

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

Second ellipsis (three dots) passed to particular functions, combining default and user-specified argument lists.

toa can be used in functions that pass argument lists separately to several functions. Internal defaults can be set per function (e.g., 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

toa(d, a, ..., quiet = FALSE)

Arguments

d Default arguments (list or vector)
a Arguments specified by user (list or vector). Can also be a single TRUE, in which case d will be returned.
... Names of unchangeable arguments (that will not be overwritten) as character strings. Can also be a vector with character 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:

```r
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:

```r
# dots passed to plot

class(a) <- c("class", "class")

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)
```

```r
testfun()
testfun(type="1", 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:

```r
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
```

packagePath

Description

Base path of package (with DESCRIPTION file), per default at current getwd. Derived from devtools::package_file

Usage

packagePath(path = ".", warnonly = FALSE)
panelDim

Arguments

path
Path to (or below) package directory. DEFAULT: "."
warnonly
Logical: if no part of the path is a package, give a warning and return the original input instead of stopping with an error. DEFAULT: FALSE

Value
Path character string

Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Sep 2017

See Also
getwd

Examples

# packagePath() # may fail on cran checks

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)

# Advanced 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
parallelCode

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

## End(Not run)

---

parallelCode  

**code chunk for parallelization**

---

**Description**

message a code chunk template for parallelization with progress bar on windows. On Linux, just use `pblapply(X, cl=8, FUN=fun)`

**Usage**

parallelCode()

**Author(s)**

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

**Examples**

parallelCode()

---

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, open = TRUE, quiet = FALSE, tracewarnmes = !quiet, 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%02d", otherwise the png will only contain the last figure. Note: this overwrites files as the % notation is not 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)

open
Logical: open file(s) after creation using openfile? DEFAULT: TRUE

quiet
Logical: suppress file creation messages and expr execution error tracing? DEFAULT: FALSE

tracewarnmes
Logical: trace warnings and messages in expr execution? Errors are always traced. Default: !quiet

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

file paths, invisible

Author(s)

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

See Also

pdf, png

Examples

## Not run: # excluded from CRAN checks, file opening not wanted
pdfpng(  plot(rnorm(500), type="l") , file="dummyplot", png=FALSE)

## End(Not run)
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 length 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
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, ...)
quantileBands

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), recycled 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

```r
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 vector 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:
qbttot(p=0.999, 2, 9) # dist with Q99.9% = 0.62
betaPlot(2, 9, cumulative=FALSE, keeppar=TRUE)
abline(h=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))
lines(density(n1000), col=3)
# with small sample size, high quantiles are systematically
# underestimated. For Q0.999, n must be > 1000

## Not run:
# Excluded from CRAN Checks because of the long computing time

# Parametrical quantiles can avoid sample size dependency!
library2("extremeStat")
library2("pbapply")
dlq <- distLOquantile(rbeta(1000, 2, 9), probs=0.999, list=TRUE, gpd=FALSE)
plotLOquantile(dlq, nbest=10) # 10 distribution functions
select <- c("wei", "wak", "pe3", "gno", "gev", "gum", "gpa", "gam")

# median of 10 simulations:
nnsim <- 10 # set higher for less noisy image (but more computing time)
qmm <- function(size, truncate=0) median(replicate(n=nnsim,
expr=quantileMean(rbeta(size, 2, 9), probs=0.999, names=FALSE,
                truncate=truncate))

pqmm <- function(size, truncate=0) median(replicate(n=nnsim,
expr=mean(distLOquantile(rbeta(size, 2, 9), probs=0.999, selection=select,
                      probbars=FALSE, time=FALSE, truncate=truncate, gpd=FALSE,
                      weighted=FALSE, empirical=FALSE, ssquiet=TRUE)[1:8, 1]))

n <- round( logSpaced(min=10, max=1000, n=15, base=1.4, plot=FALSE)  )

medians_emp <- pbapplyn(n, qmm) # medians of regular quantile average
# with truncation, only top 20% used for quantile estimation (censored quant):
medians_emp_trunc <- sapply(n, qmm, truncate=0.8)
# medians of parametrical quantile estimation
medians_param <- pbapply(n, pqmm)
# takes ~60 secs
medians_param_trunc <- pbapply(n, pqmm, truncate=0.8)

plot(n, medians_emp, type="l", ylim=c(0.45, 0.7), las=1)
abline(h=qbeta(p=0.999, 2, 9), col=6) # real value
lines(n, medians_emp_trunc, col=2) # don't help!
# In small samples, rare high values, on average, simply do not occur
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)
**round0**

**Round numbers with leading and trailing zeros**

**Description**

Round numbers and add leading + trailing zeros

**Usage**

```r
round0(x, digits = 0, pre = 2, width = digits + pre + ifelse(digits == 0, 0, 1), flag = 0, ...)
```

**Arguments**

- **x**: Value(s)
- **digits**: Number of digits (after decimal separator) to keep. DEFAULT: 0
- **pre**: Minimum number of characters before the decimal separator. DEFAULT: 2
- **width**: Total width (number of characters including dot). DEFAULT: digits + pre (+1 if needed)
- **flag**: Flag. Could be "" for spaces. DEFAULT: "0"
- **...**: Further arguments passed to `formatC`, except for "format".

**Value**

Character string vector

**Author(s)**

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

**See Also**

`formatC`, `sprintf`

**Examples**

```r
round0( pi*10^(-3:5), 2)
stopifnot(round0(17.3, 2) == "17.30")
round0(7.3)
round0(c(7.3,777.1234), 2)
round0(c(0.2,7.3,12.8), 2, pre=1)
round0(c(0.2,7.3,12.8), 1, pre=3, flag="") # spaces instead of zeros
```
roundedRect  Rectangles with rounded corners

Description

Draw rectangles with rounded corners via polygon

Usage

roundedRect(xleft, ybottom, xright, ytop, rounding = 0.25,
bothsame = TRUE, aspcorrect = bothsame, devcorrect = bothsame,
corfactor = 1.3, factorpoints = FALSE, corners = 1:4,
npoints = 200, plot = TRUE, ...)

Arguments

xleft, ybottom, xright, ytop
Single numbers with the outer end locations of the rectangle.

rounding
Proportion of the box to round. Recommended to be between 0 and 1. DEFAULT: 0.25

bothsame
Set the visual amount of rounding to the same in both x and y direction? If TRUE (the default), the proportion relates to the shortest rectangle side. This is visually correct only if aspcorrect and devcorrect are both left at TRUE and corfactor is set correctly. bothsame DEFAULT: TRUE

aspcorrect
Correct for graph aspect ratio? DEFAULT: bothsame

devcorrect
Correct for device aspect ratio? DEFAULT: bothsame

corfactor
Aspect correction factor. I found this by trial and error. More elegant solutions are welcome! DEFAULT 1.3, works well for 7x5 (width x height) graphs

factorpoints
Logical: plot points at inset locations to determine the exact value for corfactor by measuring on screen. DEFAULT: FALSE

corners
Vector with integers indicating which corners to round. Starting bottom left, going clockwise. Zero to suppress rounding. DEFAULT: 1:4

npoints
Total number of vertices for the corners. DEFAULT: 200

plot
Logical. Plot the polygon? FALSE to only compute coordinates. DEFAULT: TRUE

...
Further arguments passed to polygon, like col, border, ...

Value

Final coordinates, invisible

Author(s)

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

See Also
textField

Examples

plot(1:10) ; rect(4,2,7,8, border=8)
roundedRect(4,2,7,8, rounding=0.1)
roundedRect(4,2,7,8, rounding=0.25) # default
roundedRect(4,2,7,8, rounding=0.5)
roundedRect(4,2,7,8, rounding=-0.1, border="red")
roundedRect(4,2,7,8, rounding=1.1, border="blue")
roundedRect(2,2,8,4, rounding=0.5) # in long boxes, 0.5 is max
roundedRect(2,2,8,4, rounding=0.5, bothsame=FALSE, corfactor=1, border=3)

plot(1:10) ; rect(4,2,7,8, border=8)
roundedRect(4,2,7,8, corners=c(2,4))

plot(1:10, asp=1) ; rect(4,2,7,8, border=8)
roundedRect(4,2,7,8)
roundedRect(4,2,7,8, aspcorrect=FALSE, border="red") # results depend on asp

plot(1:10, asp=1.5) ; rect(4,2,7,8, border=8)
roundedRect(4,2,7,8)
roundedRect(4,2,7,8, aspcorrect=FALSE, border="red") # results depend on asp

plot(1:10, asp=1) ; rect(4,2,7,8, border=8)
roundedRect(4,2,7,8) # difference only visible if rect is clearly not a square:
roundedRect(4,2,7,8, bothsame=FALSE, border="red")
roundedRect(4,2,7,8, bothsame=FALSE, aspcorrect=TRUE, border="blue")

## Not run: # aspect correction factor determination
rrtest <- function(...) roundedRect(10,0.5, 35,15, border=2, factorpoints=TRUE)
pdfpng((plot(1:40
  plot(1:40, ylim=c(0,15)) ; rrtest();
  plot(1:40, ylim=c(0,15), asp=1); rrtest();
  roundedRect(2,0, 8,15, factorpoints=TRUE);
  roundedRect(15,10, 25,16, npoints=200)),
  file="dummytest", png=F, overwrite=T)

## End(Not run)
seasonality

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
seasonality

Usage

seasonality(dates, values, data, drange = NA, vrange = NA, shift = 0,
  janline = TRUE, hlines = FALSE, 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,
  ...)
seasonality

rmin  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 function 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 season, time (year) and values 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 passed 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
...
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 shiftA≠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 if plot=2, like add, format, lines.
Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Jul-Oct 2016

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

Examples

```r
# 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 the first 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)

## Not run: # excluded from CRAN checks because it is slow
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)
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**

---

**seqPal**

*Sequential color palette*

---

**Description**

Sequential color palette from yellow to blue or custom colors.

**Usage**

```r
seqPal(n = 100, reverse = FALSE, alpha = 1, extr = FALSE,
      yb = FALSE, yr = FALSE, gb = FALSE, b = FALSE, colors = NULL,
      logbase = 1, ...)
```

**Arguments**

- `n` Number of colors. DEFAULT: 100
- `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, "YlGnBu"). 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
- `b` Should colors be in an increasingly saturated blue? 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

```r
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=1.007))

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](http://r.789695.n4.nabble.com/seq-range-argument-td4684627.html)
showPal

Examples

```r
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)
# internally calls extendrange with f=extend
```

Description

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

Usage

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

Arguments

- **cex** Character 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

```r
showPal()
```
smallPlot

Inset small plot within figure

Description

multipanel-compatible inset plot with margins, background and border. Adding points after `smallPlot` is called may be incorrect if the original function messes with the graph margins, see the note in `colPointsLegend`.

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")

# select focus for further add-ons:
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)
smoothLines

draw smoothed lines

```r
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) # cannot 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) # cannot 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(1,1,1,1,2,2,3,3,2,3,3,4,4,5,5), 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

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 numerical. 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 smoothLines(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("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))
sortDF

sort dataframes by column

Description

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

Usage

sortDF(df, col, decreasing = TRUE, quiet = FALSE, ...)

Arguments

df       Data.frame to be sorted
col      Column (index or (un)quoted name) to be sorted by
decreasing Logical: should highest value be on top? DEFAULT: TRUE (unlike order!)
quiet    Logical: suppress non-df warning? DEFAULT: FALSE
...      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, insertRows, addRows

Examples

sortDF(USArrests[USArrests$Murder>11,], Assault)
sortDF(USArrests[USArrests$Murder>11,], "Assault") # safer within functions
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 character strings or strftime 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 strftime. 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("1980-01-01") + 0:5000,
  vals = cumsum(rnorm(5001)) + 50)
fakeData$vals <- fakeData$vals + sin(0:5000/365*2*pi)*max(abs(fakeData$vals))

sp <- spiralDate(time, vals, data=fakeData)
tail(sp)
spiralDate(time, vals, data=fakeData, range=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, 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\ntrends 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:
library("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, probar = 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

probar
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 sum-
m-ing to one, not just the storage cascade model.

Value

Vector of streamflow values

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, 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, 0,0,0,0,0),
    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(t, type="n", ylim=c(0, 15), xlim=c(1, 10), axes=FALSE, ann=FALSE)
axis(4, las=1)
polygon(x=c(1:11, 11:1), y=c(SPcum[1], rep(0, 11)), col=2)
for(i in 2:6) polygon(x=c(1:11, 11:1), y=c(SPcum[i], rev(SPcum[i-1])), col=i+1)
text(3.5, 1, "Shape of UH")
lines(superPos(N, UH), lwd=3)

plot(UH, type="o", ylim=lim0(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("topleft", 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("topleft", c("P a","P b", "P c"), title="with UH_2", col=3:5, lty=1)
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, 
  colargs = NULL, rowargs = NULL, mainargs = 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
- `...` Further arguments passed to all `text` like `cex`, `col`, `srt`, ...
- `classargs` List of arguments specifying how to call `classify`, e.g. method. DEFAULT: `NULL`
- `cellargs`, `colargs`, `rowargs`, `mainargs` List of arguments passed to `text` only for the cells, column labels, row labels or title, respectively. DEFAULTS: `NULL`

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`, `sortDF`
testExamples

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, namesheight=0.5, colargs=c(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()
openFile("TableColVal.pdf")
unlink("TableColVal.pdf")

## End(Not run)

testExamples

Test examples in a package

Description

Test all examples in a package

Usage

testExamples(path = packagePath("."), commentDontrun = FALSE,
  selection = NULL, logfolder = "ExampleTestLogs",
  elogfile = "errors.txt", wlogfile = "warnings.txt",
  tlogfile = "times.txt", plotfile = "plots.pdf",
  tellcurrentfile = TRUE, ...)


Arguments

- **path**: Path to package. For internal function `testExample`, path to a single Rd file. DEFAULT: `packagePath(".")`
- **commentDontrun**: Logical. Should 'dontrun sections be excluded? DEFAULT: FALSE
- **selection**: Optional: selection of files, e.g 1:10. DEFAULT: NULL
- **logfolder**: Directory where to store the logfiles. Created if not existing. DEFAULT: "ExampleTestLogs"
- **elogfile**: File to log errors in. (Appended to existing text). DEFAULT: "errors.txt"
- **wlogfile**: File to log warnings and messages in. (Appended to existing text). DEFAULT: "warnings.txt"
- **tlogfile**: File in which to write computing times. DEFAULT: "times.txt"
- **plotfile**: File to log warnings and messages in. (Appended to existing text). DEFAULT: "plots.pdf"
- **tellcurrentfile**: Logical: At the beginning of each file, message the name and current time in the console?
- **...**: Further arguments passed to internal function `testExample` and from there to `tools::Rd2ex`

Value

Logical indicating successful tests

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Mar 2019

See Also

The `evaluate` package

Examples

```r
# testExamples(selection=1:10)
```

---

**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 rounded edges.
Usage

textField(x, y, labels = seq_along(x), fill = "white", border = NA,
expression = NA, margin = 0.3, field = "rounded", nv = 500,
rounding = 0.25, rrarg = NULL, 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: 500
rounding between 0 and 0.5: portion of height that is cut off rounded at edges when field
= "rounded". DEFAULT: 0.25
rrarg List of arguments passed to roundedRect. DEFAULT: NULL
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
textField

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

s.label in package ade4, which is not so versatile and doesn’t work with logarithmic axes

Examples

```r
# TextFields with mixed field shapes
set.seed(13); plot(cumsum(rnorm(100)), type="l", main="berryFunctions::textField")
for(i in 1:2) 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(0,0.6,len=5)), field="round",
    fill=(2:6), mar=7, rounding=seq(0,0.6,len=5), border=7)
# 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 outside of the plot region (figure) into the margins:
textField(10000, -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 textField
# I'm working on it through expression, which does not work with newlines yet

---

**TTest**

*Test logical expressions*

**Description**

Check if logical expressions return what you expect with a truth table

**Usage**

```
TTest(..., 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
timeAxis

Description

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

Usage

```r
timeAxis(side = 1, timeAxis = NA, origin = "1970-01-01", startyear = NULL, stopyear = NULL, n = 5, npm = NULL, npy = NA, format = "%d.%m.%Y", yformat = "%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 for as.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.

Examples

```r
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
```
**timeAxis**

- **npy**: Number of labels per year, overrides npm and n. DEFAULT: NA
- **format**: Format of date, see details in `strptime`. DEFAULT: "%d.%m.%Y"
- **yformat**: Format of year if ym=TRUE. Use yformat=" " (with space) to suppress year labeling. DEFAULT: "%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: 0.6
- **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
%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**: Character EXPansion (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)

**Value**

The dates that were labeled

**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**

```r
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)
timeAxis(side=1)
timeAxis(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(50)), type="l", xaxt="n", ann=FALSE)

monthAxis() # see more examples there - it largely replaces timeAxis!!!

plot(Date1, cumsum(rnorm(30)), type="l", xaxt="n", ann=FALSE)
timeAxis(labels=FALSE, col.ticks=2)
timeAxis(1, format=""") # equivalent to axis(labels=FALSE)
timeAxis(1)
d <- timeAxis(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)
timeAxis(npy=12, format=""") # fix number of labels per year
timeAxis(tcl=-0.8, lwd.ticks=2, format="%Y/%m", mgp=c(3,1,0))
timeAxis(format="", mgp=c(3,2,0)) # International Date format YYYY-mm-dd

plot(Date2, cumsum(rnorm(50)), type="l", xaxt="n", ann=FALSE)
timeAxis(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)
timeAxis(1, n=4, font=2)
timeAxis(1, col.axis=3) # too many labels with default n=5

monthAxis(side=3) # again: use monthAxis, it is usually nicer!

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

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

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

---

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

```
## Not run: ## Skip time consuming checks on CRAN
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

## End(Not run)
```
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**

```r
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**

```r
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, prefix="in ", suffix=": "),
                                   "How to use traceCall in functions ", call.=FALSE); a}
upper(3)
```
Description

truncate long vectors for messages

Usage

`truncMessage(x, ntrunc = 3, prefix = "s", midfix = " ", altnix = """, sep = ", ")`

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: ""
- **sep** Character: Separator between elements. DEFAULT: ", "

Value

Character string

Author(s)

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

See Also

`message`

Examples

```r
truncMessage("hi")
message("listing name", truncMessage("hi", ", ")
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**

```r
tryStack(expr, silent = FALSE, warn = TRUE, short = TRUE, file = "", removetry = TRUE, skip = NULL)
```

**Arguments**

- **expr**: Expression 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
- **skip**: Character string(s) to be removed from the stack. e.g. "eval(expr, p)". Use short=F to find exact matches. DEFAULT: NULL

**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


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 ------

is.error( d <- upper("42"), TRUE, TRUE) # error, no d creation
traceback() # calling stack, but only in interactive mode
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")) # like try, but with traceback, even for warnings
cat(d)
d <- tryStack(upper("42"), silent=TRUE, warn=0) # don't trace warnings
d <- tryStack(upper("42"), short=FALSE)

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")
openFile("log.txt")
unlink("log.txt")

## End(Not run)

op <- options(warn=2)
d <- try(upper("42"))
cat(d)
unitHydrograph

# 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

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 absorption = 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

```r
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)
```
yearSample

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(70,40,80)[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

cats 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

```r
generateSample(2016)
# Have a nerdy
set.seed(12353); sample(0:9, 4, replace=TRUE)
```
Index

*Topic **IO**
  combineFiles, 34
  compareFiles, 35
  dupes, 42
  getName, 54
  na, 106
*Topic **aplot**
  ciBand, 18
  circle, 20
  colPoints, 26
  colPointsHist, 30
  colPointsLegend, 32
  expReg, 45
  funnelPlot, 48
  legendMt, 72
  linReg, 79
  locArrow, 81
  locLine, 82
  logAxis, 83
  logHist, 85
  logVals, 88
  monthAxis, 94
  mReg, 100
  popleaf, 119
  quantileBands, 121
  runAxis, 130
  seasonality, 131
  smallPlot, 138
  smoothLines, 140
  spiralDate, 143
  spiralDateAnim, 144
  textField, 150
  timeAxis, 154
*Topic **arith**
  approx2, 10
  logSpaced, 87
  sortDF, 142
*Topic **array**
  insertRows, 63
  l2array, 66
  panelDim, 115
*Topic **chron**
  monthAxis, 94
  monthLabs, 96
  spiralDate, 143
  spiralDateAnim, 144
  timeAxis, 154
  timer, 156
*Topic **classif**
  classify, 21
*Topic **color**
  addAlpha, 5
  addFade, 6
  catPal, 16
  colPoints, 26
  colPointsHist, 30
  colPointsLegend, 32
  divPal, 41
  rainbow2, 125
  seqPal, 135
  showPal, 137
  spiralDate, 143
  spiralDateAnim, 144
*Topic **datagen**
  seqR, 136
*Topic **distribution**
<table>
<thead>
<tr>
<th>Topic</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>berryFunctions-package</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>createFun</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>dataStr</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>extime</td>
<td>46</td>
</tr>
<tr>
<td>dplot</td>
<td>addAlpha</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>addFade</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>approx2</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>catPal</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>divPal</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>groupHist</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>lin0</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>linLogHist</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>linLogTrans</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>logAxis</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>logHist</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>logVals</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>monthAxis</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>panelDim</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>pretty2</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>quantileBands</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>rainbow2</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>seqPal</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>showPal</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>smallPlot</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td>timeAxis</td>
<td>154</td>
</tr>
<tr>
<td>dynamic</td>
<td>linLogHist</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>linLogTrans</td>
<td>77</td>
</tr>
<tr>
<td>error</td>
<td>is.error</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>traceCall</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>tryStack</td>
<td>161</td>
</tr>
<tr>
<td>file</td>
<td>checkFile</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>combineFiles</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>compareFiles</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>createPres</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>dupes</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>learnVocab</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>lsMem</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>na9</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>newFilename</td>
<td>108</td>
</tr>
<tr>
<td>hplot</td>
<td>betaPlot</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>betaPlotComp</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>ciBand</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>climateGraph</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>colPoints</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>expReg</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>funnelPlot</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>groupHist</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>horizHist</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>linLogHist</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>linLogTrans</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>linReg</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>lsc</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>mReg</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>normPlot</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>smallPlot</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td>spiralDate</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>spiralDateAnim</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>superPos</td>
<td>146</td>
</tr>
<tr>
<td></td>
<td>tableColVal</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>unitHydrograph</td>
<td>163</td>
</tr>
<tr>
<td>iplot</td>
<td>locArrow</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>locLine</td>
<td>82</td>
</tr>
<tr>
<td>list</td>
<td>l2array</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>12df</td>
<td>69</td>
</tr>
<tr>
<td>logic</td>
<td>between</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Ttest</td>
<td>153</td>
</tr>
<tr>
<td>manipulate</td>
<td>headtail</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>insertRows</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>l2array</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>12df</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>movAv</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>movAvLines</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>rescale</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>sortDF</td>
<td>142</td>
</tr>
<tr>
<td>misc</td>
<td>addRows</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>insertRows</td>
<td>63</td>
</tr>
<tr>
<td>multivariate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data, 39</td>
<td>install.packages, 64, 73</td>
<td></td>
</tr>
<tr>
<td>data.frame, 7</td>
<td>instGit, 64</td>
<td></td>
</tr>
<tr>
<td>datastr, 39</td>
<td>is.character, 51</td>
<td></td>
</tr>
<tr>
<td>Date, 94</td>
<td>is.error, 65</td>
<td></td>
</tr>
<tr>
<td>dbeta, 12–14</td>
<td>kge (gof), 55</td>
<td></td>
</tr>
<tr>
<td>decompose, 98</td>
<td>l2array, 66, 69</td>
<td></td>
</tr>
<tr>
<td>density, 33</td>
<td>l2df, 67, 68</td>
<td></td>
</tr>
<tr>
<td>distance, 40</td>
<td>learnVocab, 71</td>
<td></td>
</tr>
<tr>
<td>divPal, 16, 41, 135, 137</td>
<td>legend, 44, 45, 48, 72, 80, 81, 102, 113</td>
<td></td>
</tr>
<tr>
<td>dnorm, 111</td>
<td>legendmt, 72</td>
<td></td>
</tr>
<tr>
<td>dupes, 36, 42</td>
<td>letters, 107, 165</td>
<td></td>
</tr>
<tr>
<td>eval.parent, 118</td>
<td>library, 73</td>
<td></td>
</tr>
<tr>
<td>example, 47</td>
<td>library2, 73</td>
<td></td>
</tr>
<tr>
<td>exp4p, 43, 101</td>
<td>lim0, 74</td>
<td></td>
</tr>
<tr>
<td>expReg, 45, 81</td>
<td>lines, 12, 100, 111, 132, 141</td>
<td></td>
</tr>
<tr>
<td>extendrange, 74, 136</td>
<td>linLogHist, 75, 77, 145</td>
<td></td>
</tr>
<tr>
<td>exTime, 46</td>
<td>linLogTrans, 76, 77</td>
<td></td>
</tr>
<tr>
<td>file.exists, 17, 109</td>
<td>linReg, 46, 79</td>
<td></td>
</tr>
<tr>
<td>filter, 98</td>
<td>lm, 44–46, 56, 80, 81, 103</td>
<td></td>
</tr>
<tr>
<td>findInterval, 15</td>
<td>locArrow, 81, 83</td>
<td></td>
</tr>
<tr>
<td>format, 84, 88</td>
<td>locator, 82, 83</td>
<td></td>
</tr>
<tr>
<td>formatC, 128</td>
<td>locLine, 82, 82</td>
<td></td>
</tr>
<tr>
<td>funnelPlot, 48</td>
<td>loess, 98</td>
<td></td>
</tr>
<tr>
<td>funSource, 9, 38, 51, 65</td>
<td>log, 87</td>
<td></td>
</tr>
<tr>
<td>gc, 55</td>
<td>log10, 85, 87, 89</td>
<td></td>
</tr>
<tr>
<td>gofNA (gof), 55</td>
<td>logAxis, 45, 76, 83, 86, 88, 89, 131</td>
<td></td>
</tr>
<tr>
<td>googleLink2pdf, 57</td>
<td>logHist, 85</td>
<td></td>
</tr>
<tr>
<td>groupHist, 59, 116</td>
<td>logical, 154</td>
<td></td>
</tr>
<tr>
<td>gsub, 37, 58, 126</td>
<td>logSpaced, 21, 87, 135</td>
<td></td>
</tr>
<tr>
<td>head, 36, 60</td>
<td>logVals, 75, 77, 78, 83–85, 88, 121</td>
<td></td>
</tr>
<tr>
<td>headTail, 60</td>
<td>ls, 93</td>
<td></td>
</tr>
<tr>
<td>heatmap, 148</td>
<td>lsc, 90, 146, 164</td>
<td></td>
</tr>
<tr>
<td>help, 47, 67</td>
<td>lsMem, 93</td>
<td></td>
</tr>
<tr>
<td>hist, 31, 59, 61, 62, 75, 76, 86</td>
<td>matrix, 7</td>
<td></td>
</tr>
<tr>
<td>horizHist, 61</td>
<td>mean, 121, 122</td>
<td></td>
</tr>
<tr>
<td>help, 47, 67</td>
<td>median, 121, 122</td>
<td></td>
</tr>
<tr>
<td>hist, 31, 59, 61, 62, 75, 76, 86</td>
<td>message, 39, 47, 66, 73, 113, 160, 161</td>
<td></td>
</tr>
<tr>
<td>horizHist, 61</td>
<td>min, 15</td>
<td></td>
</tr>
<tr>
<td>iconv, 37</td>
<td>model.frame, 101</td>
<td></td>
</tr>
<tr>
<td>image, 27</td>
<td>monthAxis, 94, 131</td>
<td></td>
</tr>
<tr>
<td>inherits, 66</td>
<td>monthLabs, 95, 96, 155</td>
<td></td>
</tr>
<tr>
<td>insertRows, 7, 63, 142</td>
<td>movAv, 97, 100, 121</td>
<td></td>
</tr>
<tr>
<td>na9, 106</td>
<td>movAvLines, 98, 99</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mReg, 44, 46, 81, 100</td>
<td></td>
</tr>
</tbody>
</table>
INDEX

nameSample, 107, 165
newFilename, 34, 38, 108, 110, 118
nndist, 40
normalizePath, 109, 110
normalizePathCP, 109
normplot, 13, 110
nse, 91
nse(gof), 55

object.size, 93
openFile, 38, 112, 118
optim, 44, 55, 101, 103
options, 88
order, 142
owa, 113, 132

packagePath, 37, 114, 150
panelDim, 115
par, 12, 14, 19, 28, 31, 33, 44, 46, 48, 59, 74, 75, 77, 80, 81, 84, 94, 101, 111, 133, 137, 138, 154, 155
parallelCode, 117
paste, 35, 97, 106, 120
pdf, 47, 117, 118, 148
pdfpng, 117
plot, 12, 14, 19, 28, 44, 46, 48, 81, 87, 90, 111, 113
plot.default, 74
png, 117, 118
points, 18, 28
polygon, 12, 18–20, 24, 45, 111, 122, 129
popleaf, 119
POSIXct, 94, 154
predict.lm, 45, 46, 80
pretty, 120, 121, 154
pretty2, 120
prettyNum, 88
quantile, 121–123
quantileBands, 19, 121
quantileMean, 122, 123, 132, 133
rainbow, 125
rainbow2, 125
range, 120, 136
rbind, 7
Rd2ex, 150
read.table, 106
readLines, 36
rect, 129
removeSpace, 126
require2(library2), 73
rescale, 127
rgb, 5, 141
rmse, 91
rmse(gof), 55
rollapply, 98
round0, 128
roundedapply, 129, 151, 152
Rprofile, 71
rsquare(gof), 55
runAxis, 130
sample, 107, 165
sapply, 69
scan, 34, 35
seasonality, 131, 144
segments, 27, 28
seq, 87, 136
seqPal, 16, 27, 32, 42, 125, 135, 137, 148
seqR, 80, 136
set.seed, 107, 118, 165
setwd, 38
showNonASCII, 37
showPal, 16, 42, 135, 137
signif, 82
sin, 20
smallPlot, 27, 28, 31–33, 138
smooth, 98
smoothLines, 140
sort, 142
sortDF, 7, 12, 64, 69, 142, 148
spiralDate, 132–134, 143, 145
spiralDateAnim, 143, 144
split, 59
sprintf, 128
stop, 17, 66
str, 39
strheight, 138
strptime, 94, 132, 143, 155
strsplit, 58
strwidth, 33, 151
sub, 126
subplot, 139
subset, 52
substitute, 28, 52, 54
substr, 158
superPos, 91, 146
suppressMessages, 47
suppressWarnings, 47
symbols, 20
Syntax, 153
Sys.sleep, 75, 77, 145, 157
system.time, 47
system2, 9, 37, 38, 112
tableColVal, 147
tail, 60
tapply, 59
tempfile, 47
testExamples, 149
text, 33, 122, 148, 151, 152
textField, 14, 130, 133, 150
TFTest, 153
timeAxis, 95, 97, 154
timer, 156
title, 133, 138
toupper, 158
toupper1, 158
traceback, 159
traceCall, 159, 162
truncMessage, 160
try, 66, 161, 162
tryCatch, 161
tryStack, 159, 161
txtProgressBar, 157
unitHydrograph, 91, 146, 163
View, 11, 12
warning, 17, 161
which, 11
write, 35
xy.coords, 45, 80
yearSample, 107, 165