Package: smwrGraphs (via r-universe)

October 24, 2024

Version 1.1.4.9000

Date 2019-08-07

Title Graphing Functions

Depends smwrBase (>= 1.0.0), methods

Imports KernSmooth, akima, lubridate

Suggests smwrData (>= 1.0.0), dataRetrieval, knitr, rmarkdown, captioner

Description Functions to create high-quality graphs to support statistical methods in water resources. These graphs meet or nearly meet the publication standards of the U.S. Geological Survey.

BugReports https://github.com/USGS-R/smwrGraphs/issues

URL https://pubs.er.usgs.gov/publication/ofr20161188

License CC0

LazyLoad yes

LazyData yes

RoxygenNote 6.1.1

Repository https://ldecicco-usgs.r-universe.dev

RemoteUrl https://github.com/USGS-R/smwrGraphs

RemoteRef HEAD

RemoteSha ef457c36c39fc527e3da15d73fb5ff1b7e183bb7

Contents

`smwrGraphs-package` *Graphing Functions*

Description

Functions to create high-quality graphs.

These graphs meet or nearly meet the publication standards for illustrations of the U.S. Geological Survey (USGS) (U.S. Geological Survey, written commun., 2012). They are intended to be a suite of integrated functions that make producing graphs and figures relatively easy by passing much information about the plots between functions so the user does not need to manage graphical information.

Details

Package:	smwrGraphs
Type:	Package
License:	CC0
Depends:	smwrBase (>= 1.0.0), methods
Imports:	KernSmooth, akima, lubridate
Suggests:	smwrData (>= 1.0.0), dataRetrieval

The functions in the `smwrGraphs` package are an integrated suite of functions that facilitate the production of graphs that nearly meet USGS publication standards for illustrations (U.S. Geological Survey, written commun., 2012). Those standards include line weight, tick placement, labels, font size, and layout of the explanation. The font used in production very closely matches the standard Univers Condensed, and was selected because of its broad availability on many computer platforms.

Use of base `R` or other graphics functions can result in inconsistent lineweights, font sizes and styles, and can require manual manipulation of the explanation. The Programmer's Guide section in Lorenz (2015) shows examples of calls to lower level graphics functions in base `R` that produce consistent graphics products.

Functions to set up and initialize the `smwrGraphs` environment:
`preSurface`
`setGD`
`setGraph`
`setKnitr`
`setLayout`
`setPDF`
`setPage`
`setPNG`
`setRStudio`
`setRtMargin`
`setSplom`
`setSweave`
`setTopMargin`

Main plotting functions:
`areaPlot`
`biPlot`
`boxPlot`
`colorPlot`
`condition`
`contourPlot`
`corGram`
`dendGram`
`dotPlot`
`ecdfPlot`
`histGram`

piperPlot
probPlot
qqPlot
reportGraph
scalePlot
seasonPlot
splomPlot
seriesPlot
stiffPlot
surfacePlot
ternaryPlot
timePlot
transPlot
xyPlot
condition

Functions to add features to a plot:
addAnnotation
addArea
addAxisLabels
addCaption
addCI
addErrorBars
addExplanation
addGrid
addLabel
addMinorTicks
addPiper
addSLR
addSmooth
addStiff
addTable
addTernary
addTitle
addXY
labelPoints
refLine

Data Manipulation Functions for Graphs:
cov2Ellipse
dataEllipse
hull
interpLine
paraSpline

Color palettes:
blueRed.colors
coolWarm.colors

greenRed.colors
pastel.colors
redBlue.colors
redGreen.colors
warmCool.colors

Selected Miscellaneous Functions:
copyDemo
strip.blanks

Author(s)

Dave Lorenz

References

Lorenz, D.L., Diekoff, A.L, smwrGraphs—an R package for graphing hydrologic data, version 1.1.2.
U.S. Geological Survey, 2012, Author's guide to standards for U.S. Geological Survey page-size illustrations, 37 p. https://pubs.er.usgs.gov/publication/ofr20161188

Examples

```
# For these examples, print to console
.pager <- options("pager")
options(pager="console")
# See the demo for examples of how to use the functions in this library.
demo(package="smwrGraphs")
# A simple listing of the vignettes in this package:
vignette(package="smwrGraphs")
options(.pager)
```

---

addAnnotation *Add Text to a Graph*

Description

Adds text to a plot to annotate a feature.

Usage

```
addAnnotation(x, y, annotation, leaderx = NULL, leadery = NULL,
  leadercol = "black", angle = 0, justification = "left", size = 60
  * par("csi"), position = "above", current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE))
```

Arguments

x	the x-axis placement of annotation.
y	the y-axis placement of annotation.
annotation	the text. Can be either a character string or an "expression" object.
leaderx	draw leader from x to leaderx.
leadery	draw leader from y to leadery.
leadercol	the color of the leader.
angle	the angle to rotate the text.
justification	the justification of the text relative to x, y. Must be one of "left," "center," or "right."
size	size of the text in points, the default is the current point size.
position	the vertical location of the text. Must be one of "above," "below," or "center."
current	the current plot controls. Typically, this would be the output from one of the graph creation functions like xyPlot.

Details

This function places only a single annotation string on the graph for each call.
A leader from x, y to leaderx, leadery if leaderx is not NULL.

Value

The current plot information is returned invisibly.

See Also

labelPoints, addTable, xyPlot

Examples

```
## Not run:
set.seed(12)
X <- rnorm(12)
Y <- X + rnorm(12)
# make an outlier
X[1L] <- 1.5
setGD()
AA.pl <- xyPlot(X, Y)
# label the outlier
addAnnotation(X[1L], Y[1L], "Outlier", current=AA.pl)
# For more details of addAnnotation see
vignette(topic="GraphAdditions", package="smwrGraphs")
demo(topic="TopAxisExample", package="smwrGraphs")

## End(Not run)
```

addArea *Add a Filled Polygon to Graph*

Description

Adds a filled polygon (area) to a graph.

Usage

```
addArea(x, y, ybase = NULL, Area = list(name = "", color = "gray",
  outline = "black"), current = list(yaxis.log = FALSE, yaxis.rev =
  FALSE, xaxis.log = FALSE))
```

Arguments

x	the x-axis coordinates of the polygon. Missing values are not permitted.
y	the y-axis coordinates of the polygon. Missing values are not permitted.
ybase	the y-axis coordinates of the polygon. See **Details**. Missing values are not permitted.
Area	parameters defining the characteristics of the area. See areaPlot for details.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot.

Details

If ybase is NULL, then x and y should form a complete polygon, which can be closed or open. Otherwise, ybase can be a single value in which case the area between ybase and y is treated as the area, or ybase can be a vector as long as y and the area between is treated as the area to be shaded.

Value

The current plot information is returned invisibly.

See Also

areaPlot, addXY, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- seq(1, 9, by=.5)
Y <- runif(17) + runif(17)
setGD()
AA.pl <- xyPlot(X, Y, Plot=list(what="none"))
addArea(X, Y, ybase=0, current=AA.pl)
# For more details of addArea see
```

```
vignette(topic="GraphSetup", package="smwrGraphs")
demo(topic="DurationHydrograph", package="smwrGraphs")

## End(Not run)
```

---

addAxisLabels *Axis Ticks and Labels*

Description

Adds axis ticks, labels, and title to a graph.

Usage

```
addAxisLabels(which, current, title = "", ticks = FALSE,
  labels = TRUE)
```

Arguments

which	which axis to label, must be one of "bottom," "left," "top," or "right."
current	a list containing the current plot information, see **Details**.
title	the axis title.
ticks	draw the ticks.
labels	draw the labels.

Details

The current argument is generally the output from a high-level plotting function in smwrGraphs. If which is "left" or "right," then current must contain a component named yax. If which is "bottom" or "top," then current must containa component named xax. Those components are generally constructed from functions like linearPretty or or logPretty.

Value

The current plot information is returned invisibly.

Note

In general, all functions that create plots will draw the necessary axes. This function should be used only to add axis labels to an unlabeled axis. Axis labels can be suppressed by setting up the margins with negative values.

See Also

linearPretty, logPretty, datePretty, transPretty, addLabel

Examples

```
## Not run:
set.seed(1)
X <- runif(25)
Y <- runif(25)
AA.pl <- xyPlot(X, Y)
addAxisLabels("top", AA.pl, labels=TRUE)
# For more details of addAxisLabels see
vignette(topic="GraphSetup", package="smwrGraphs")

## End(Not run)
```

---

addBars *Bar Graph*

Description

Creates a bar chart by adding bars to an existing graph.

Usage

```
addBars(x, y, base = 0, Bars = list(name = "Auto", fill = "gray80",
  outline = "black", width = "Auto", orientation = "stack"),
  current = list(yaxis.log = FALSE, yaxis.rev = FALSE, xaxis.log =
  FALSE))
```

Arguments

x	the x-coordinate data. Missing values are permitted but result in no bar.
y	the heights of the bars or y-coordinate data. Missing values are permitted but result in no bar. For stacked or grouped bars, y must be a matrix, each row corresponding to the value in x and the columns representing each bar. See **Note**.
base	extend the bars from this value.
Bars	parameters defining the characteristics of the bars. See **Details**.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot. See **Note**.

Details

The Bars argument must be a tagged list with these components:

name a character vector describing each column of data; used in the explanation. If "Auto," then derive the name from the column names in y.

fill the name of the color to fill each bar. For multiple bars, can be a vector of colors or the name of a color sequence generating function, such as "pastel.colors."

outline the name of the color to draw the outline or border for each bar. If "none," then no border is drawn.

width the width of each bar in x-axis units. For discrete x-axis, If `width` 1, then the bars form a continuous filled area. The default is "Auto," which fills 2/3 of the distance. If 0, then draw vertical lines rather than bars; the color of the line is based on `outline`.

orientation the orientation of the bars. Must be either "stack" or "group." Can be abbreviated to a single letter.

Value

The current plot information is returned invisibly.

Note

Use of `addBars` adds 1 step to creating bar charts, but adds flexibility in axis formatted from existing high-level plotting functions such as `xyPlot` or `timePlot`.

Bars are only valid for linear y-axes. Calling `addBars` when `yaxis.log` or `yaxis.rev` is `TRUE` or for any arbitrary transform of the y-axis will cause `addBars` to fail.

Datasets containing grouped data are often stacked with a column indicating the grouping. There are several functions that will reformat stacked datasets. The `group2row` function is very flexible in accepting many types of data to reformat rather than only numeric data.

See Also

`xyPlot`, `timePlot`, `addXY` `group2row`

Examples

```
## Not run:
set.seed(1)
X <- seq(1, 9, by=1.0)
Y <- runif(9) + runif(9)
setGD()
AA.pl <- xyPlot(X, Y, Plot=list(what="none"), yaxis.range=c(0,2))
addBars(X, Y, base=0, current=AA.pl)
# For more details of addBars see
demo(topic="AnnualFlowBarChart", package="smwrGraphs")

## End(Not run)
```

---

addCaption *Add Caption*

Description

Adds a caption at the bottom of the graph.

Usage

```
addCaption(caption = "")
```

Arguments

caption	the text of the caption for the graph

Value

Nothing is returned.

Note

Useful for adding 1-line captions.

Examples

```
## Not run:
set.seed(1)
X <- runif(25)
Y <- runif(25)
setGD()
AA.pl <- xyPlot(X, Y)
addCaption("Twenty five random points")
# See for examples of addCaption:
vignette(topic="GraphSetup", package="smwrGraphs")
demo(topic="AnnualFlowBarChart", package="smwrGraphs")

## End(Not run)
```

---

addCI *Add Confidence Interval Lines*

Description

Adds confidence interval lines to a simple linear regression or q-normal graph.

Usage

```
addCI(type, level = 0.95, Plot = list(name = "", what = "lines", type =
  "solid", width = "standard", color = "black"), current = list(yaxis.log
  = FALSE, yaxis.rev = FALSE, xaxis.log = FALSE))
```

Arguments

type	the type of confidence interval desired. Must be either "SLR" for the confidence interval for a simple linear regression model, which must have been created using the addSLR function or "q-norm" for the confidence interval for a q-normal plot created using qqPlot.
level	the confidence level desired.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
current	the current plot information, normally the output from a high-level graphics function like qqPlot or xyPlot.

Value

The current plot information, the x and y components are the data, not the line. The regression model is included as the lm component.

Note

The equation for the confidence intervals for a simple linear regression model can be found in any textbook on regression, see section 9.4.4 in Helsel and Hirsch (2002) for example.
The confidence interval for a normal distribution is described in Appendix 9 in U.S. Water Resources Council (1982). Owen (1968) describes the application of the noncentral *t*-distribution for computing the tolerance limits for a normal distribition.

References

Helsel, D.R., and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water-Resources Investigations, book 4, chap. A3, 522 p.
U.S. Water Resources Council, 1982, Guidelines for determining flood flow frequency, revised September 1981, Editorial Corrections March 1982: Hydrology Committee Bulletin 17B, Washington D.C., 190 p., 1 plate.

See Also

addSLR, qqPlot, addErrorBars

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
setGD()
AA.pl <- qqPlot(X)
addCI("q-norm", current=AA.pl)
# For more details of addCI see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

addErrorBars *Errors Bars*

Description

Adds upper-lower error bars to data in a graph.

Usage

```
addErrorBars(x, yup, ylo, Bars = list(name = "", cap = 0.09, width =
  "standard", color = "black"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE))
```

Arguments

x	the x-coordinate data. Missing values are permitted but result in no bar.
yup	the upper limit of the error bar. Missing values are permitted but result in no bar.
ylo	the lower limit of the error bar. Missing values are permitted but result in no bar.
Bars	parameters defining the characteristics of the error bars. See **Details**.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot.

Details

The Bars argument must be a tagged list with these components:

name a name describing the data; used in the explanation.

cap the width of each cap on the error bar.

width the width of the lines drawn for the error bars.

color the name of the color to draw the error bars.

Value

The current plot information is returned invisibly.

Note

The error bars are plotted on top of any current symbol. To plot the symbol on top of the error bar, start with the argument Plot=list(what="none") in the original call to xyPlot or timePlot and then add the symbols with a call to addXY. Note that this is only necessary if the color of the symbol and the color of the error bars are different.
The symbol drawn for errors bars in the explantion does not have caps due to a limitation in the system for creating the explanation.

See Also

xyPlot, timePlot, addXY, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- seq(1, 9, by=1.0)
Y1 <- runif(9)
Y2 <- runif(9)
Y <- (Y1 + Y2)/2
Ymin <- pmin(Y1, Y2)
Ymax <- pmax(Y1, Y2)
setGD()
AA.pl <- xyPlot(X, Y, yaxis.range=c(0,1))
addErrorBars(X,Ymax, Ymin)
# For more details of addErrorBars see
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

`addExplanation` *Add Explanation*

Description

Creates or adds an explanation, also called key or legend.

Usage

```
addExplanation(what, where = "new",
  title = expression(bold(EXPLANATION)), box.off = where != "new",
  margin = rep(0, 4), line.length = 2, line.height = 1.1)
```

Arguments

`what`	a specialized object for an explanation, from the output from calls to the plotting functions
`where`	a description of where to put the explanation, see **Details**.
`title`	the title of the explanation.
`box.off`	logical, if `TRUE`, then box off the explanation with a blank background and black box, otherwise the background is not blanked and no bounding box is drawn.
`margin`	the margin for a new graph
`line.length`	the relative length of lines drawn in the explanation, see **Details**.
`line.height`	the relative spacing of the lines in the explanation.

Details

The value for `where` must be one of "ul," "ur," "ll," "lr," "cl," "cr," "uc," "lc," "cc," or "new." If "new," then the explanation is placed in a new graph, otherwise, the first letter is an abbreviation for upper, lower, or center and the second letter is an abbreviation for left, right, or center. the explanation for a box plot, stiff diagram, or contour plot must be placed in a new graph. If `box.off` is `TRUE`, then the explanation abuts the axes, otherwise it is placed slightly inset so that the text does not interfere with the ticks.

In most cases, `line.length` does not need to be changed. In some cases, such as mass produced figures that will not be modified by an illustrator, the `line.length` can be increased to show full dashes if dashed lines are drawn. In gneralt, the illustrator should create dashed lines rather than drawing them in the graph.

Value

Nothing is returned.

Note

The call to `addExplanation` should be the last in any sequence of calls to construct a figure becuase it can alter some graphical parameters.
Box plot explanations require fairly large graph areas because of the detail required for some types. In general, a graph about 4.5 inches high is needed for the Tukey type and 4 inches for other types and widths of 2.5 and 2 inches respectively. The sizes are smaller for the font type of "USGS." If the graph area is smaller than required for the box plot explanation, then either a modified explanation is created or a warning is printed and the explanation may be unreadable.

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y, Plot=list(name="Random Points"))
addExplanation(AA.pl, where='ul')
# For more details of addExplanation see
vignette(topic="BoxPlots", package="smwrGraphs")
vignette(topic="GraphAdditions", package="smwrGraphs")
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="LineScatter", package="smwrGraphs")
vignette(topic="PiperPlot", package="smwrGraphs")
demo(topic="Coplot-complexScatterPlot", package="smwrGraphs")
demo(topic="FlowDur-Measurements", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")
demo(topic="RightAxisExample", package="smwrGraphs")

## End(Not run)
```

addGrid *Grid Lines*

Description

Adds grid lines to a graph.

Usage

```
addGrid(current, Xgrid = list(grid = "gray50", finegrid = "none"),
  Ygrid = list(grid = "gray50", finegrid = "none"))
```

Arguments

current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot. See **Details**.
Xgrid	parameters defining the characteristics of the x-axis grid lines. The components refer to the color to draw the grid (at ticks) or finegrid (between ticks).
Ygrid	parameters defining the characteristics of the y-axis grid lines. The components refer to the color to draw the grid (at ticks) or finerid (between ticks).

Details

Information about grid lines is contained in the information returned from high-level plotting functions in the smwrGraphs package.

Value

NULL is returned invisibly.

Note

The function addGrid should be used after setting up a graph with one of the main plotting functions in the smwrGraphs package and setting the what component in the Plot argument to "none." The graph can be completed by using addXY.

See Also

xyPlot, timePlot, addXY

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y, Plot=list(what="none"))
```

```
# Grid first, then data to avoid over plotting
addGrid(AA.pl)
addXY(X, Y, Plot=list(what="points"))
# For more details of addGrid see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

---

addLabel *Axis Labels*

Description

Adds text in the margin for specialized axis labels.

Usage

```
addLabel(label, x, side = "bottom", size = "Auto", distance = 0.2,
  justification = "center", orientation = "parallel",
  current = list(yaxis.log = FALSE, yaxis.rev = FALSE, xaxis.log =
  FALSE))
```

Arguments

label	the text or expression to add to the graph.
x	the axis location in the correct user units.
side	the side to place label. Must be "bottom," "left," "top," or "right." Only the first letter is necessary.
size	the size of the text in points.
distance	the distance from the axis, in lines of text.
justification	defines the placement of the text relative to x if orientation is "parallel" and relative to distance if orientation is "perpendicular." Must be one of "left," "center," or "right."
orientation	the orientation of the label relative to the axis. Must be either "parallel" or "perpendicular."
current	the current plot parameters. Typically, this would be the output from one of the graph creation functions like xyPlot.

Value

Nothing is returned.

Note

In general, all functions that create plots will draw the necessary axes. This function should be used only to add axis labels to an unlabeled axis. Axis labels can be suppressed by setting up the margins with negative values or, for some functions, special arguments to xlabels or ylabels.

See Also

addAxisLabels, mtext and plotmath for example expressions,

Examples

```
## Not run:
set.seed(1)
X <- as.POSIXct(c("2010-12-22 10:30", "2010-12-28 13:45",
"2011-01-05 9:30", "2011-01-07 14:50"))
Y <- runif(4)
setGD()
AA.pl <- timePlot(X, Y, Plot=list(what="points"))
# Insert verical bar between years
addLabel("|", as.Date("2011-01-01"), distance=1.2)
# For more details of addLabel see
vignette(topic="GraphGallery", package="smwrGraphs")
demo(topic="AnnualFlowBarChart", package="smwrGraphs")

## End(Not run)
```

---

addMinorTicks *Add Axis Ticks*

Description

Adds minor axis ticks to a graph.

Usage

```
addMinorTicks(which, current, ticks)
```

Arguments

which	which axis to label, must be one of "bottom," "left," "top," or "right," "x," or "y." If which is "x," then add both bottom and top minors ticks. If which is "y," then add both left and right minor ticks.
current	the current plot information, see **Details**.
ticks	the number of minor ticks to draw. If missing, then the default number is used, see **Details**.

Details

The current argument must contain a component named yax if which is "left" or "right" or a component named xax if which is "bottom" or "top." Those arguments are generally constructed from functions like linearPretty.

The default placement of minor ticks is at the largest unit that lies between the major ticks—if the difference between major ticks is an even multiple of 2 or 5, then the minor interval will be that even multiple of 1, otherwise it will be 1/10 that even multiple.

Value

The current plot information is returned invisibly.

Note

In general, this should be used only with linear axes. Other axis types can result in unexpected results.

See Also

linearPretty, addAxisLabels, addLabel

Examples

```
## Not run:
set.seed(1)
X <- runif(25, .5, 9.5)
Y <- runif(25)
setGD()
AA.pl <- xyPlot(X, Y)
addMinorTicks("bottom", AA.pl)
addMinorTicks("top", AA.pl)
# For more details of addMinorTicks see
vignette(topic="DateAxisFormats", package="smwrGraphs")

## End(Not run)
```

---

addPiper *Add Detail to a Piper Plot*

Description

Adds points or lines to a Piper plot.

Usage

```
addPiper(xCat, yCat, zCat, xAn, yAn, zAn, xPip, yPip, Plot = list(name =
  "", what = "points", type = "solid", width = "standard", symbol =
  "circle", filled = TRUE, size = 0.09, color = "black"),
  current = list())
```

Arguments

xCat	,
yCat	, and
zCat	the cations for the x-, y-, and z-axes. Need not sum to 1 or 100. See **Details**.
xAn	,

yAn	, and
zAn	the anions for the x-, y-, and z-axes. Need not sum to 1 or 100. See **Details**.
xPip	, and
yPip	the coordinates for the internal piper diagram. See **Details**.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like piperPlot.

Details

The values for xCat, yCat, zCat, xAn, yAn, and zAn should match the constituents in the original call to piperPlot. The units should be in milli-equivalents if all three components are used to add to the Piper plot.

There are two ways to add to a Piper plot, the first way is to specify new data by supplying values for xCat, yCat, zCat, xAn, yAn, and zAn. In this case the data for the central middle (Piper) plot are generated from those data and the xPip, yPip arguments will be ignored. The other way is to use processed values from an already created Piper plot to add details to the plot, either customized symbols or closed lines showing groups for example. The vignette PiperPlot shows an example of customized symbols.

Value

Information about the graph.

See Also

piperPlot

Examples

```
## Not run:
# See for examples of addPiper:
vignette(topic="PiperPlot", package="smwrGraphs")

## End(Not run)
```

---

addSLR *Add a Regression Line*

Description

Adds a simple linear regression line to a scatter plot.

Usage

```
addSLR(x, y, Plot = list(name = "", what = "lines", type = "solid", width
  = "standard", color = "black"), Model = list(x = "", y = "", form =
  "exp", where = "none"), current = list(yaxis.log = FALSE, yaxis.rev =
  FALSE, xaxis.log = FALSE), ...)

## Default S3 method:
addSLR(x, y, Plot = list(name = "", what = "lines",
  type = "solid", width = "standard", color = "black"), Model = list(x =
  "", y = "", form = "exp", where = "none"), current = list(yaxis.log =
  FALSE, yaxis.rev = FALSE, xaxis.log = FALSE), ...)

## S3 method for class 'list'
addSLR(x, y, Plot = list(name = "", what = "lines", type =
  "solid", width = "standard", color = "black"), Model = list(x = "", y =
  "", form = "exp", where = "none"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE), ...)
```

Arguments

x	the x-axis data. For method list, x is a list that contains components x and y and the y argument is not used. Missing values are permitted but omitted in the regression.
y	the y-axis data. Missing values are permitted but omitted.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
Model	parameters for displaying the simple linear regression model. See **Details**.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot.
...	not used, required for other methods.

Details

The Model argument sets up the information to include the regression model equation on the graphs. Within Model, x is the name to use for the explanatory variable, y is the name to use for the response variable, form indicates the form that the regression model should take if the variable are log transformed; "exp" indicates that the model equation should be expressed as an exponent, any other string indicates thet the model should be expressed using the transformation functions; and where indicates where to place the equation. The value for where is a two letter code based on "upper," "center," or "lower" and "right," "center," or "left"—for example "ul" would place the model equaiton in the upper left corner.

Value

The current plot information, the x and y components are the data, not the line. The regression model is included as the lm component.

See Also

addXY, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y)
addSLR(AA.pl)
# For more details of addSLR see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

---

addSmooth *Add a Smooth Line*

Description

Adds a smoothed line to a scatter plot.

Usage

```
addSmooth(x, y, Smooth = "loess.smooth", ..., Smooth.along = "x",
  Plot = list(name = "", what = "lines", type = "solid", width =
  "standard", color = "black"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE))

## Default S3 method:
addSmooth(x, y, Smooth = "loess.smooth", ...,
  Smooth.along = "x", Plot = list(name = "", what = "lines", type =
  "solid", width = "standard", color = "black"), current = list(yaxis.log
  = FALSE, yaxis.rev = FALSE, xaxis.log = FALSE))

## S3 method for class 'list'
addSmooth(x, y, Smooth = "loess.smooth", ...,
  Smooth.along = "x", Plot = list(name = "", what = "lines", type =
  "solid", width = "standard", color = "black"), current = list(yaxis.log
  = FALSE, yaxis.rev = FALSE, xaxis.log = FALSE))
```

Arguments

x	the x-axis data. For method `list`, x is a list that contains components x and y and the y argument is not used. Missing values are permitted and ignored in the smooth.

y	the y-axis data. Missing values are permitted and ignored in the smooth.
Smooth	the name of the smoothing function. See **Details**.
...	additional parameters for the function names in Smooth.
Smooth.along	the data along which the smoother is run. Must be either "x," which smooths y along x resulting in a horzontal line, or "y," which smooths x along y resulting in a vertical line.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot.

Details

The value for Smooth must the name of a smoothing function as a character string. The default is "loess.smooth" but any smoother that accepts arguments names x and y and other arguments controlling the smooth and returns a list with components named x and y can be used. Examples of other smoothers in base R are "supsmu" and "smooth.spline."

Value

The current plot information.

Note

If an error is generated from the smoother, then nothing is added to the graph, an error is printed, the returned object contains missing values for the data that should have been plotted, and the explanation is not updated.

See Also

addXY, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y)
addSmooth(AA.pl)
# For more details of addSmooth see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

addTable *Add Table to Graph*

Description

Adds a small table to a graph.

Usage

```
addTable(tbl, where = "ll", title = "")
```

Arguments

tbl	the data frame or matrix to add to graph. All data must be of mode character, which allows the user to format the data rather than the automatic formatting done by R.
where	character specifying the corner the table should be placed, see **Details**.
title	the title of the table.

Details

where must be one of "ul," "ur," "ll," or "lr." The first letter is an abbreviation for upper or lower, the second letter is an abbreviation for left or right.

Value

Nothing is returned.

Note

Each column of the table can be formatted by the user, using the format function, to control the alingment of the data in each column of the table. The column names of the table are the column names of the matrix or data frame. If there are no column names in a matrix, then the table is printed without column names. A matrix gives the user more control over column names than does a data frame.

See Also

addExplanation, addAnnotation

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y)
```

```
Mat <- cbind(c("Mean of X", "Mean of Y"), round(c(mean(X), mean(Y)), 2))
addTable(Mat, "ul")
# For more details of addTable see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

---

addTernary *Add Detail to a Ternary Diagram*

Description

Adds points or lines to a Ternary diagram.

Usage

```
addTernary(x, y, z, Plot = list(name = "", what = "points", type =
  "solid", width = "standard", symbol = "circle", filled = TRUE, size =
  0.09, color = "black"), current = list())
```

Arguments

x	the x-axis (bottom) data. Missing values are permitted, but result in breaks in the plotted data.
y	the y-axis (left side) data. Missing values are permitted, but result in breaks in the plotted data.
z	the z-axis (right side) data. Note that x, y, and z do not need to sum to the axis range. Missing values are permitted, but result in breaks in the plotted data.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like ternaryPlot.

Value

Information about the graph.

See Also

ternaryPlot

Examples

```
## Not run:
# See for examples of addTernary:
vignette(topic="PiperPlot", package="smwrGraphs")

## End(Not run)
```

addTitle *Add a Title*

Description

Adds a title (also called heading) to a graph.

Usage

```
addTitle(Main = "", Heading = "", Justification = "left",
  Bold = TRUE, Position = "above")
```

Arguments

Main	the main text of the title. Can be either a character string or an "expression" object. If Main is an "expression" object, then Heading and Bold will be ignored.
Heading	The title heading, generally a single letter. See **Details**
Justification	specify the horizontal location of the title, must be one of "left," "center," or "right."
Bold	logical, if TRUE, then display the title in bold face type.
Position	specify the vertical location of the title, must be either "above" or "inside."

Details

If only Heading is non blank, then the title is a single letter in bold italics. If both Heading and Main are non blank, then the title is a single letter followed by a period in bold italics followed by Main in bold if Bold is TRUE.

Value

Nothing is returned.

See Also

addCaption, addAnnotation, addTable

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y)
addTitle("X and Y")
# For more details of addTitle see
vignette(topic="BoxPlots", package="smwrGraphs")
```

```
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")

## End(Not run)
```

---

addXY *Add a plot to a graph*

Description

Adds points or lines to the current graph.

Usage

```
addXY(x, y, ...)

## S4 method for signature 'ANY,numeric'
addXY(x, y, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE), new.axis = "none",
  new.log = FALSE, new.rev = FALSE, new.range = c(NA, NA),
  new.labels = 7, new.title = "")

## S4 method for signature 'numeric,character'
addXY(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE), jitter.y = FALSE)
```

Arguments

x	the x-axis data. Missing values are permitted, but result in breaks in the plotted data.
y	the y-axis data. Missing values are permitted, but result in breaks in the plotted data.
...	arguments for specific methods.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
current	the current plot information. Typically, this would be the output from one of the graph creation functions like xyPlot.
new.axis	character: indicating which new axis to set up. Must be either "right," "top," or "none," which indicates that the existing axes be used (default).
new.log	logical, if TRUE, then log transform new axis.
new.rev	logical, if TRUE, then reverse new axis.

new.range	set new-axis range.
new.labels	set up new-axis labels.
new.title	the new-axis title.
jitter.y	adjust y values to reduce overlap for each group?

Value

Information about the graph.

Methods

signature(x ="ANY", y = "numeric" Any valid x-axis data and numeric y.

signature(x ="numeric", y = "character" Method to add to a dot plot; the right-axis arguments are not valid.

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
Y2 <- X + rnorm(32, sd=0.5)
setGD()
AA.pl <- xyPlot(X, Y)
addXY(X, Y2, Plot=list(what="points", color="brown"))
# See for examples of addXY:
vignette(topic="GraphAdditions", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="ProbabilityPlots", package="smwrGraphs")
demo(topic="DurationHydrograph", package="smwrGraphs")
demo(topic="FlowDur-Measurements", package="smwrGraphs")
demo(topic="MeasurementRating", package="smwrGraphs")
demo(topic="RightAxisExample", package="smwrGraphs")
demo(topic="TopAxisExample", package="smwrGraphs")

## End(Not run)
```

---

areaPlot *Shaded Area Plot*

Description

Produces a plot where the area between lines is filled with color.

Usage

```
areaPlot(x, y, Areas = list(name = "Auto", fillDir = "between", base =
  "Auto", lineColor = "black", fillColors = "pastel"), yaxis.log = FALSE,
  yaxis.range = c(NA, NA), xaxis.log = FALSE, xaxis.range = c(NA,
  NA), ylabels = 7, xlabels = "Auto", xtitle = "", ytitle = "",
  caption = "", margin = c(NA, NA, NA, NA))
```

Arguments

x	numeric x-axis coordinates in increasing order.
y	a numeric matrix of y-axis coordinates.
Areas	parameters controlling the areas. See **Details**.
yaxis.log	logical, if TRUE, then log transform y axis.
yaxis.range	set y-axis range. See **Details**.
xaxis.log	logical, if TRUE, then log transform x axis.
xaxis.range	set x-axis range. See **Details**.
ylabels	set up y-axis labels. See linearPretty for details.
xlabels	set up x-axis labels. See linearPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the caption for the graph.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.

Details

The components of Areas are name, the name or names to use to describe the areas in the explanation, the default "Auto" generates names from the column names of y; fillDir, how to fill—must be either "between" or "under;" base, the base value when fillDir is "under," can be "Auto" to draw to the x-axis or any numeric value; lineColor specifies the color to draw the lines around each area, may be "none" for no line drawn; fillColors specifies colors for each area, when only a single area is drawn, then the value must be the name of a color, otherwise either a vector of colornames or the prefix name of a function that generates a sequence of colors. The prefix name is prepended to ".colors" for the name of the function. See the documentation for these functions in the **See Also** section.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

See Also

addArea, smwr.colors, heat.colors

Examples

```
## Not run:
set.seed(1)
X <- seq(1, 9, by=.5)
Y <- runif(17) + runif(17)
setGD()
# The default fillDir, between, requires at least a 2-column matrix
areaPlot(X, cbind(rep(0, 17),Y))
# For more details of areaPlot see
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

biPlot *Biplot*

Description

Produces a biplot, which is a plot of two different types of data on the same graph.

Usage

```
biPlot(x, ...)
```

Arguments

x	any object which has a valid method for biPlot.
...	additional arguments for other methods.

Value

Information about the graph

Note

A call should be made to setPage to set up the graphics environment before calling biPlot.

See Also

setPage, biPlot.default, biPlot.princomp

Examples

```
## Not run:
# See for examples of biPlot:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

`biPlot.default` *Biplot*

Description

Produces a biplot, which is a plot of two different types of data on the same graph.

Usage

```
## Default S3 method:
biPlot(x, y, separate.axes = TRUE, xPlot = list(name
  = "observations", what = "points", type = "solid", width = "standard",
  symbol = "circle", filled = TRUE, size = 0.05, color = "black"),
  yPlot = list(name = "variables", width = "color", size = 0.2, color =
  "darkblue", symbol = "arrow", filled = FALSE),
  xPlotLabels = list(labels = "rownames", dir = "NE", size = 8, offset =
  0.35), yPlotLabels = list(labels = "colnames", dir = "Auto", size = 8,
  offset = 0.35, color = "darkblue"), ylabels = 5, xlabels = 5,
  ylabels2 = 5, xlabels2 = 5, xtitle = "", ytitle = "",
  xtitle2 = "", ytitle2 = "", range.factor = 1.25, caption = "",
  margin = c(NA, NA, NA, NA), ...)
```

Arguments

`x`	a 2-column matrix of x- (column 1) and y- (column 2) coordinates for observations or equivalent.
`y`	a 2-column matrix of x- (column 1) and y- (column 2) coordinates for variables or equivalent.
`separate.axes`	logical, if `TRUE`, then plot `x` and `y` data on separate axes.
`xPlot`	control information to plot the `x` data. See `setPlot` for a description of the parameters.
`yPlot`	control information to plot the `y` data. See `setPlot` for a description of the parameters. For `yPlot`, symbol can be "arrow" to indicate that an arrow is to be drawn from the origin.
`xPlotLabels`	control information for the `x` data labels. See **Details**.
`yPlotLabels`	control information for the `y` data labels. See **Details**.
`ylabels`	set y-axis labels for `x` data. See `linearPretty` for details.

xlabels	set x-axis labels for x data. See linearPretty for details.
ylabels2	set y-axis labels for y data. See linearPretty for details.
xlabels2	set x-axis labels for y data. See linearPretty for details.
xtitle	x-axis title (also called x-axis caption) for x data.
ytitle	y-axis title (also called y-axis caption) for x data.
xtitle2	x-axis title (also called x-axis caption) for y data.
ytitle2	y-axis title (also called y-axis caption) for y data.
range.factor	a numeric factor by which to expand the axis ranges so that labels can be drawn.
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	not used, required for other methods.

Details

The xPlotLabels and yPlotLabels arguments must be tagged lists with these components:

labels the labels. For xPlotLabels, "rownames" means use the row names from x to generate the labels. For yPlotLabels, "colnames" means use the column names from y to generate the labels. Otherwise a character vector of the labels.

dir the direction the label text is placed from the object.

size the size of the label text.

offset the distance the labels is placed relative to the object.

color the color of the label text.

Value

Information about the graph

Note

A call should be made to setPage to set up the graphics environment before calling biPlot.

See Also

setPage, biPlot

biPlot.princomp *Biplot*

Description

Produces a biplot, which is a plot of two different types of data on the same graph, from a principal component analysis.

Usage

```
## S3 method for class 'princomp'
biPlot(x, Which = 1:2, Scale = "Auto",
  obsPlot = list(name = "observations", what = "points", type = "solid",
  width = "standard", symbol = "circle", filled = TRUE, size = 0.05, color
  = "black"), varPlot = list(name = "variables", width = "color", size =
  0.2, color = "darkblue", symbol = "arrow", filled = FALSE),
  obsPlotLabels = list(labels = "rownames", dir = "NE", size = 8, offset
  = 0.75), varPlotLabels = list(labels = "colnames", dir = "Auto", size =
  8, offset = 0.75, color = "darkblue"), ylabels = 5, xlabels = 5,
  ylabels2 = "Auto", xlabels2 = "Auto", xtitle = "Auto",
  ytitle = "Auto", xtitle2 = "Auto", ytitle2 = "Auto",
  range.factor = 1.25, caption = "", margin = c(NA, NA, NA, NA), ...)

## S3 method for class 'prcomp'
biPlot(x, Which = 1:2, Scale = "Auto",
  obsPlot = list(name = "observations", what = "points", type = "solid",
  width = "standard", symbol = "circle", filled = TRUE, size = 0.05, color
  = "black"), varPlot = list(name = "variables", width = "color", size =
  0.2, color = "darkblue", symbol = "arrow", filled = FALSE),
  obsPlotLabels = list(labels = "rownames", dir = "NE", size = 8, offset
  = 0.75), varPlotLabels = list(labels = "colnames", dir = "Auto", size =
  8, offset = 0.75, color = "darkblue"), ylabels = 5, xlabels = 5,
  ylabels2 = "Auto", xlabels2 = "Auto", xtitle = "Auto",
  ytitle = "Auto", xtitle2 = "Auto", ytitle2 = "Auto",
  range.factor = 1.25, caption = "", margin = c(NA, NA, NA, NA), ...)
```

Arguments

x	an object of class "princomp" that has the information to create a biplot.
Which	sequence of two numbers indicating which components to plot.
Scale	either a character string indicating the scaling option between observations and variables, or numeric value controling the scaling. If character, then must be one of "auto," "distance," "symmetric," "variance," or "interpolative." See **Details**.
obsPlot	control information to plot the observations. See setPlot for a description of the parameters.

varPlot	control information to plot the variables. See setPlot for a description of the parameters. For yPlot, symbol can be "arrow" to indicate that an arrow is to be drawn from the origin.
obsPlotLabels	control information for the observation labels. See **Details**.
varPlotLabels	control information for the variable labels. See **Details**.
ylabels	set y-axis labels for the observation data.
xlabels	set x-axis labels for the observation data.
ylabels2	set y-axis labels for the variable data.
xlabels2	set x-axis labels for the variable data.
xtitle	x-axis title (also called x-axis caption) for the observation data.
ytitle	y-axis title (also called y-axis caption) for the observation data.
xtitle2	x-axis title (also called x-axis caption) for the variable data.
ytitle2	y-axis title (also called y-axis caption) for the variable data.
range.factor	a numeric factor by which to expand the axis ranges so that labels can be drawn.
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	not used, required for other methods.

Details

The scaling between observations and variables is controlled by Scale, which can take any value between 0 and 1 or a character string indicating a specific scaling. The options for the character string are: "distance," which produces a plot where the observations retain their approximate relation with respect to Eucldiean distances and correponds to a numeric value of 1; "variance," which produces a plot where the cosine of the angle between the variable vectors is related to the correlation between the variables and corresponds to a numeric value of 0; "symmetric," which tries to balance the range of values for observations and variables to give a pleasing graph and corresponds to a numeric value of 0.5; or "Auto," which is the same as "variance." Another option for Scale is "interpolative," which produces a specialized axis scaling so that the approximate values of the variables can be obtained for each observation. It is not implemented in this version.

The obsPlotLabels and varPlotLabels arguments must be tagged lists with these components:

labels the labels. For xPlotLabels, "rownames" means use the row names from x to generate the labels. For yPlotLabels, "colnames" means use the column names from y to generate the labels. Otherwise a character vector of the labels.

dir the direction the label text is placed from the object.

size the size of the label text.

offset the distance the labels is placed relative to the object.

color the color of the label text.

Value

Information about the graph.

Note

A call should be made to `setPage` to set up the graphics environment before calling `biPlot`.

References

Gower, J.C. and Hand, D.J., 1996, Biplots, Chapman and Hall, London, 277 p.

See Also

`setPage`, `biPlot`

Examples

```
## Not run:
# See for examples of biPlot:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

boxPlot *Box Plot*

Description

Produces a truncated, simple, Tukey, or extended box plot.

Usage

```
boxPlot(..., group = NULL, Box = list(type = "truncated", show.counts =
  TRUE, nobox = 5, width = "Auto", fill = "none", truncated = c(10, 90)),
  yaxis.log = FALSE, yaxis.range = c(NA, NA), ylabels = "Auto",
  xlabels = "Auto", xlabels.rotate = FALSE, xtitle = "",
  ytitle = "", caption = "", margin = c(NA, NA, NA, NA))

## S3 method for class 'numeric'
boxPlot(..., group = NULL, Box = list(type =
  "truncated", show.counts = TRUE, nobox = 5, width = "Auto", fill =
  "none", truncated = c(10, 90)), yaxis.log = FALSE,
  yaxis.range = c(NA, NA), ylabels = "Auto", xlabels = "Auto",
  xlabels.rotate = FALSE, xtitle = "", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA))

## S3 method for class 'list'
boxPlot(..., group = NULL, Box = list(type =
```

```
  "truncated", show.counts = TRUE, nobox = 5, width = "Auto", fill =
  "none", truncated = c(10, 90)), yaxis.log = FALSE,
  yaxis.range = c(NA, NA), ylabels = "Auto", xlabels = "Auto",
  xlabels.rotate = FALSE, xtitle = "", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA))

## S3 method for class 'data.frame'
boxPlot(..., group = NULL, Box = list(type =
  "truncated", show.counts = TRUE, nobox = 5, width = "Auto", fill =
  "none", truncated = c(10, 90)), yaxis.log = FALSE,
  yaxis.range = c(NA, NA), ylabels = "Auto", xlabels = "Auto",
  xlabels.rotate = FALSE, xtitle = "", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA))
```

Arguments

`...`	the data to plot. Missing values are permitted and excluded from the summary statistics computations.
`group`	any vector containing distinct values to create groups of data for individual box plots. Missing values are not permitted. Valid only when a single numeric vector is supplied for `...`.
`Box`	control parameters for the box. See **Details**.
`yaxis.log`	logical, if `TRUE`, then log transform y axis.
`yaxis.range`	set y-axis range. See **Details**.
`ylabels`	set up y-axis labels. See `linearPretty` for details; the value for `ylabels` can be set to an valid value for the `label` argument in `linearPretty` or a tagged list with values spcified for the arguments in `linearPretty`.
`xlabels`	set up x-axis labels. Must be either "Auto" or a character vector of the x-axis labels.
`xlabels.rotate`	logical, if `TRUE`, then rotate x-axis labels 90 degrees.
`xtitle`	the x-axis title (also called x-axis caption).
`ytitle`	the y-axis title (also called y-axis caption).
`caption`	the figure caption
`margin`	set the plot area margins, in units of lines of text. Generally all NA or the output from `setGraph` if appropriate.

Details

If group is numeric, then the boxes will be plotted along a continuous numeric axis. Otherwise the x-axis will be discrete groups.
`Box` is a list with these components:

type the type of boxtplot: "simple" the whiskers extend to the minimum and maximum of the data, "truncated" the whiskers extend to percentiles defined by `truncated`, "tukey" the standard "Tukey" boxplot as described by Helsel and Hirsch (200), and "extended" the whisker extend to percentiles defined by `truncated` and values outside of that range are shown;

show.counts show the number of observations used to compute the boxplot statistics;

nobox only individual values are shown if the number of observations is less than or equal to this value;

width the width of the box, in inches;

fill The color of the filled box or "none" for no fill;

truncated the percentiles to use for the truncated boxplot.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choices should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to `setPage` to set up the graphics environment before calling `boxPlot`.
If `yaxis.log` is set to `TRUE`, then the quartiles and interquartile range are computed from the log-transformed values rather then the untransformed values, which is common for other box plots. Those computations are in agreement with the box plots generated in the QWGRAPH component of the QWDATA module in the National Water Information System (NWIS) described by Dennis Helsel's 1989 Branch of Systems Analysis Technical Memorandum No. 89.01, available online at https://water.usgs.gov/admin/memo/BSA/BSA89.01.pdf. Those computations have a significant effect on the appearance the whiskers and outside values of the Tukey box plot and are motivated by the general assumption of a log-normal distribution for most water-quality constituents.

References

Helsel, D.R. and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water-Resources Investigations, book 4, chap. A3, 522 p.

See Also

setPage, dotPlot

Examples

```
## Not run:
set.seed(1)
Xbig <- rnorm(100)
setGD()
# The simple type box plot
boxPlot(Xbig, Box=list(type="simple"))
# For more details of boxPlot see
vignette(topic="BoxPlots", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
demo(topic="Coplot-simpleBoxPlot", package="smwrGraphs")
```

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

---

boxPlotStats *Compute Statistcs for a Box Plot*

Description

Computes the statistics for truncated, simple, Tukey, or extended box plots (support function for boxPlot).

Usage

```
boxPlotStats(x, Box, yaxis.log)
```

Arguments

x	a list containing the data to compute the statistics.
Box	control parameters for the box as set by the user and verified by boxPlot.
yaxis.log	logical, if TRUE, then log transform data before computing the statistics.

Value

a list containing the statistics for each box plot.

See Also

boxPlot

---

colorPlot *Plot Data*

Description

Produces a line/scatter plot where each point or group of related points has a unique color or where sections along a line have different colors.

Usage

```
colorPlot(x, y, color, Plot = list(), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = c(NA, NA), ylabels = 7, xlabels = 7, xtitle = "",
  ytitle = "", caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'numeric,numeric'
colorPlot(x, y, color, Plot = list(name =
  "Auto", what = "points", symbol = "circle", filled = TRUE, size = 0.09,
  color = "Auto", groups = 4, ramp = "greenRed"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = c(NA, NA), ylabels = 7, xlabels = 7,
  xtitle = deparse(substitute(x)), ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'Date,numeric'
colorPlot(x, y, color, Plot = list(name =
  "Auto", what = "lines", symbol = "circle", filled = TRUE, size = 0.09,
  color = "Auto", groups = 10, ramp = "greenRed"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = range(x, na.rm = TRUE), ylabels = 7,
  xlabels = "Auto", xtitle = "", ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'POSIXt,numeric'
colorPlot(x, y, color, Plot = list(name =
  "Auto", what = "lines", symbol = "circle", filled = TRUE, size = 0.09,
  color = "Auto", groups = 10, ramp = "greenRed"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = range(x, na.rm = TRUE), ylabels = 7,
  xlabels = "Auto", xtitle = "", ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)
```

Arguments

x	the x-axis data
y	the y-axis data
color	the colors or a class to set colors, must match the length of x and y.
Plot	tagged list of control parameters of the plot: name="Auto" means derive class names from the argument color, otherwise, must be a tagged list of color=name, ... (in which case the color tag is not used); what can be only "points" or "lines" in the current version; color="Auto" means if the argument color is double or dateLike create groups of classes, otherwise create unique colors, alternate values are "Range" (treat like double), tagged list of group_name=color, and so forth, "Discrete" valid only for numeric, or "Index" valid only for integer or for specified colors. No usable explanation is generated when color is set to "Index"—use repeated calls to addXY if an explanation is needed and specific colors are supplied.

yaxis.log	logical, if TRUE, then log-transform the y axis
yaxis.rev	logical, if TRUE, then reverse the y axis.
yaxis.range	set the range of the y-axis. See **Details**.
xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis. See **Details**.
ylabels	set up y-axis labels.
xlabels	set up x-axis labels.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	arguments for specific methods.

Details

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Methods

signature(x = "numeric", y = "numeric")) Typically used to create a colored scatter plot for numeric x and y data.

signature(x = "Date", y = "numeric") Can be used to create a hydrograph where the line is colored by a third variable, or a colored scatter plot over time.

signature(x = "POSIXt", y = "numeric") Can be used to create a hydrograph where the line is colored by a third variable, or a colored scatter plot over time.

Note

A call should be made to setPage to set up the graphics environment before calling colorPlot.

See Also

setPage, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
Z <- sqrt(X^2 + Y^2) # distance from origin
setGD()
# All defaults: color ramp from Z
colorPlot(X, Y, Z)
# See for examples of colorPlot:
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

`condition` *Conditioned Graphs*

Description

Facilitates producing a series of graphs conditioned by a grouping variable.

Usage

```
condition(plot.call, data, group, format = "grid", num.cols = NA,
  num.rows = NA, explanation = NULL, share = "", group.name = "",
  xtitle = "", ytitle = "", caption = "")
```

Arguments

`plot.call`	either a simple call to a graphics function or a sequence of calls enclosed in curly braces({})
`data`	the data.frame containing the variables used in `plot.call` and `group`.
`group`	a character string identifying the grouping variable in `data`.
`format`	the orientation of the graphs. If "table," then the graphs are created beginning in the upper-left hand corner. If "grid," then the graphs are created beginning in the lower-left See **Details**. hand corner.
`num.cols`	the number of columns on each page. See **Details**.
`num.rows`	the number of rows on each page. See **Details**.
`explanation`	where to place an explanation if necessary. See `setLayout` for details.
`share`	character, if `share` contains "x", then the code in `plot.call` is set up to share x-axes; if `share` contains "y", then the code in `plot.call` is set up to share y-axes. The default is not to share either axes.
`group.name`	a character string to prepend to each value in `group` to create the graph title.
`xtitle`	the x-axis title (also called x-axis caption).
`ytitle`	the y-axis title (also called y-axis caption).
`caption`	the figure caption.

Details

Graphs are created by executing `plot.call` for each unique value in the variable specified by `group`. The columns and rows are set by `num.cols` and `num.rows`. If both are set to missing `NA`, then each page contains 1 graph, and the graphics setup should provide for mulitple pages. If one of `num.cols` or `num.rows` is set to a numeric value, the other is calculated from the number of graphs so that all graphs would be displayed on a single page. If both `num.cols` and `num.rows` are set to numeric values, then each page would contain a maximum of `num.cols` times `num.rows` and multiple pages would be needed if that were less than the total number of graphs.

The order of the graphs is controlled by the type of `group`. If `group` is a factor, then the order is set by the order of the levels, otherwise the order is set by the order from `unique`.

Value

The value returned by `plot.call` is returned invisibly. If used correctly, it could be used to add an explanation.

Note

This function is designed to facilitate the production of conditioned graphs, but may not render each collection completely. For example, graphs arranged in table format will not have x-axis labels for incomplete columns.

A call must be made to `setPage` or `setPDF` to set up the graphics page before calling `condition`. The returned value can be used to create an explanation, if desired. If `explanation` is not `NULL`, then the graph is set to the explanation and there is no need to call `setGraph` to set up the explanation.

The called plotting function must set the `margin` argument to `.margin`. See the demos for examples.

References

Cleveland, W.S., 1993, Visualizing data: Summit, New Jersey, Hobart Press, 360 p.

See Also

`setLayout`, `setPage`, `setPDF`, `unique`

Examples

```
## Not run:
# See for examples of condition:
demo(topic="Coplot-complexScatterPlot", package="smwrGraphs")
demo(topic="Coplot-simpleBoxPlot", package="smwrGraphs")

## End(Not run)
```

contourPlot *Contour Plot*

Description

Produces a contour plot or a colored surface with colors corresponding to values in z.

Usage

```
contourPlot(z, ...)

## Default S3 method:
contourPlot(z, x, y, Grid = list(method =
  "interpolate", linear = TRUE, extrapolate = FALSE, density = 90, span =
  0.75, degree = 1, family = "symmetric"), Contours = list(name = "Auto",
  levels = 10, filled = FALSE, lineColor = "black", lineLabel = "flattest",
  fillColors = "coolWarm"), yaxis.range = c(NA, NA),
  xaxis.range = c(NA, NA), ylabels = 4, xlabels = 4,
  xtitle = deparse(substitute(x)), ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S3 method for class 'matrix'
contourPlot(z, rows, cols, matrix.rows = "x",
  Contours = list(name = "Auto", levels = 10, filled = FALSE, lineColor =
  "black", lineLabel = "flattest", fillColors = "coolWarm"),
  yaxis.range = c(NA, NA), xaxis.range = c(NA, NA), ylabels = 4,
  xlabels = 4, xtitle = deparse(substitute(x)),
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA), ...)
```

Arguments

z	the values representing the surface data.
...	not used, required for other methods.
x	the x-axis coordinates for each value in z.
y	the y-axis coordinates for each value in z.
Grid	control parameters for gridding irregularly spaced data. See **Details**.
Contours	control parameters for the coutour lines or levels in the filled plot. See **Details**.
yaxis.range	set the range of the y-axis.
xaxis.range	set the range of the x-axis.
ylabels	set up y-axis labels. See linearPretty for details.
xlabels	set up x-axis labels. See linearPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).

caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
rows	the coordinates for z represented by the rows in the matrix.
cols	the coordinates for z represented by the columns in the matrix.
matrix.rows	a single character, either "x" or "y" indicating whether the rows in z should be plotted along the x or y axis.

Details

Missing values are permitted in z, x, and y for the default method and are removed, with a warning. before constructing the surface. Duplicated values, identical x and y, are not permitted and generate an error.

Missing values are not permitted in rows or columns but are permitted in z for the matrix method. Missing values in z result in blank areas in the plot.

The Grid argument must be a tagged list with these components:

method The method to use for constructung the grid. Must be either "interpolate" or "loess." If "interpolate," then the z values are interpolated directly from the x and y values. If "loess," then the z values are smoothed prior to interpolation

linear Logical, if TRUE, then use linear interpolation, if FALSE, then use spline interpolation.

extrapolate Logical, if TRUE, then extrapolate to the limits of the grid, if FALSE, then do not extrapoalte outsite of the hull of the data values.

density The density of the grid—the number of cells along x and y.

span The span argument for loess if method is "loess."

degree The degree argument for loess if method is "loess."

family The family argument for loess if method is "loess."

#'The Contour argument must be a tagged list with these components:

name The name to use to describe the contours. If "Auto" and filled is TRUE, then the descripotion is blank. If "Auto" and filled is FALSE, the the description is Line of equal value." In all other cases, the description is the text assigned to name.

levels Either the number of levels of contours or a vector of the desired contour levels.

filled Logical, if TRUE, then draw filled contours, if FALSE, then only contour lines are drawn.

lineColor The color to draw the contour lines. Can be set to "none" to supress drawing lines for filled contours.

lineLabel A character string indicating how to draw the labels on the contours. May be "none" to supress draing the labels, or any valid value for the method argument to contour.

fillColors The prefix corresponding to a color ramp generating function, like "warmCool" for the warmCool.colors function.

Value

Information about the graph.

Examples

```
## Not run:
set.seed(1)
Xbig <- runif(100)
Ybig <- runif(100)
# Make a hill
Zbig <- 1 - ((Xbig-.5)^2 + (Ybig-.5)^2)^.75
setGD()
contourPlot(Zbig, Xbig, Ybig)
# See for examples of contourPlot see
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

copyDemo *Copy a Demo File*

Description

Copies a demo file from the source package to a file.

Usage

```
copyDemo(topic, package = "smwrGraphs", file)
```

Arguments

topic	the name of the topic, must be a character string.
package	the package name, must be a character string.
file	the target file name, must be a character string. If missing, then the file name is created from the topic name.

Value

Either the name of the target file or NULL if the copy failed.

See Also

demo

Examples

```
## Not run: copyDemo("HydroPrecip")
```

corGram *Correlogram*

Description

Creates a correlogram for irregularly spaced data.

Usage

```
corGram(x, y, Plot = list(name = "Standardized Observations", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.03, color = "gray40"), CorGram = list(band = 0.15,
  kernel = "normal", color = "black", width = "standard", add0line = TRUE),
  yaxis.range = c(-3, 3), xaxis.range = c(0, 1.5), ylabels = 7,
  xlabels = 4, xtitle = "Difference in Time, in Years",
  ytitle = "Standardized Serial Correlation", caption = "",
  margin = c(NA, NA, NA, NA))
```

Arguments

x	decimal time.
y	residuals or other observations, these will be scaled, but not centered.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
CorGram	control parameters of the correlogram line. See **Details**.
yaxis.range	set the y-axis range.
xaxis.range	set the x-axis range.
ylabels	set the y-axis labels. See linearPretty for details.
xlabels	set the x-axis labels. See linearPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.

Details

CorGram is a list with these components:

band a measure of the bandwidth used by locpoly

kernel the kernel used, currently ignored

color the color of the line representing the correlogram

width the line width of the line representing the correlogram

add0line logical, if TRUE, then add the 0 line; if FALSE, then do not draw the 0 line

Value

Information about the graph.

Note

A call should be made to `setPage` to set up the graphics environment before calling `corGram`.

See Also

`setPage`

Examples

```
## Not run:
# See for examples of corGram:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

`cov2Ellipse` *Construct an Ellipse*

Description

Constructs an ellipse from a covariance matrix.

Usage

```
cov2Ellipse(cov, center, scale = 1, n = 151)
```

Arguments

`cov`	the 2-dimenstional covariance matrix, representing x and y.
`center`	the means of x and y.
`scale`	the size of the ellipse in units of standard deviation.
`n`	the number of points in the returned data.

Value

A list containing the x- and y-coordinates of the ellipse.

See Also

`dataEllipse`

Examples

```
# make a few points on a unit circle
TMP <- cov2Ellipse(matrix(c(1,0,0,1), ncol=2), c(0,0), n=5)
# Pretty print the data
lapply(TMP, zapsmall)
```

`dataEllipse` *Construct an Ellipse*

Description

Constructs an ellipse from x- and y-coordinate data.

Usage

```
dataEllipse(x, y, percent = 100, smooth = 0)
```

Arguments

`x`	the x-coordinate data.
`y`	the y-coordinate data.
`percent`	a scale factor, adjusted to include `percent` of the data.
`smooth`	required for naming compatibility with other functions, not used.

Value

A list containing the x- and y-coordinates of the ellipse.

See Also

`cov2Ellipse`, `hull`

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
TMP <- dataEllipse(X, Y)
# Just print the first 10 values
lapply(TMP, function(x) x[1:10])
# For examples of dataEllipse in graphs see:
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

`datePretty` *Pretty Axis*

Description

Constructs information for making a nicely formatted date axis. A support function for creating date/time axes. Understanding the arguments can help in building specially formatted axes.

Usage

```
datePretty(x, major = "Auto", minor = "Auto", tick.span = 1,
  style = c("Auto", "at", "between"), label.abbr = 0)
```

Arguments

`x`	date and time data
`major`	the major tick interval, must be one of "hours," "days," "months," "years," "water years," or "Auto," which will make an intelligent choice from those. "Auto" will also automatically change `tick.span`. Abbreviations are permitted. See **Details**.
`minor`	the minor tick interval, must be one of "min," "hours," "days," "months," "years," or "Auto," which will make an intelligent choice from those, possibly adjusted by a scaling factor. Can also be formatted as an argument to `seq`, such as "15 mins" for 15-minute ticks. Abbreviations are not permitted to account for the scaling factor.
`tick.span`	span between major labels. For example, with "years" option, tick.span=5 would generate labels like 1990, 1995, 2000 and so forth.
`style`	the style of labels, must be one of "at," "between," or "Auto," which selects the "best" style. "At" places the labels at the ticks and "between" places the labels between the major ticks. Abbreviations are permitted.
`label.abbr`	indicator of the degree of abbreviation for labels

0	best guess based on number of intervals
1	shortest (single letter month, for example)
2	USGS abbreviation (3- or 4-letter month name, for example)
3	full text

Details

Setting `major` to "water year" is practical only for periods of time from 1 to 5 water years in length; also sets the date range to water years.

Value

Formatting information about the axis labels.

See Also

timePlot, month.USGS, timePretty

---

dendGram *Tree Graphs*

Description

Produces a tree graph from a hierarchical cluster analysis.

Usage

```
dendGram(x, Tree = list(orientation = "vertical", width = "standard",
  color = "black"), axis.range = c(NA, NA), labels = "Auto",
  ytitle = "", xtitle = "", caption = "", margin = c(NA, NA, NA,
  NA))
```

Arguments

x	the data to plot. Must be able to be converted to class "dendrogram."
Tree	control parameters of the tree diagram. See **Details**.
axis.range	set the range of the tree-axis.
labels	set the tree-axis labels. See linearPretty for details.
ytitle	the y-axis title (also called y-axis caption).
xtitle	the x-axis title (also called x-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.

Details

Tree is a list with these components:

orientation the orientation of the dendrogram, must be either "vertical" or "horizontal"

width the line width of the line representing the dendrogram

color the color of the line representing the dendrogram

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling ecdfPlot.

See Also

setPage, probPlot

Examples

```
## Not run:
# For an example of a dendGram  see
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

dotPlot *Dot Plot*

Description

Creates a dot plot.

Usage

```
dotPlot(x, y, Plot = list(), yaxis.orient = "", yaxis.order = "",
  yaxis.grid = TRUE, xaxis.log = FALSE, xaxis.range = c(NA, NA),
  ylabels = "", xlabels = 7, xtitle = "", ytitle = "",
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'numeric'
dotPlot(x, y, Plot = list(name = "", what = "points",
  type = "solid", width = "standard", symbol = "circle", filled = TRUE,
  size = 0.09, color = "black"), yaxis.orient = "table",
  yaxis.order = "none", yaxis.grid = TRUE, xaxis.log = FALSE,
  xaxis.range = c(NA, NA), ylabels = "full", xlabels = 7,
  xtitle = deparse(substitute(x)), ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA), jitter.y = TRUE, ...)

## S4 method for signature 'Date'
dotPlot(x, y, Plot = list(name = "", what = "points",
  type = "solid", width = "standard", symbol = "circle", filled = TRUE,
  size = 0.09, color = "black"), yaxis.orient = "table",
  yaxis.order = "none", yaxis.grid = TRUE, xaxis.log = FALSE,
  xaxis.range = range(x, na.rm = TRUE), ylabels = "full",
  xlabels = "Auto", xtitle = "", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA), jitter.y = TRUE, ...)
```

Arguments

x	the x-axis data. Missing values are permitted and not plotted.

y	the y-axis data, expected to be be either character or factor. Missing values are permitted and removed before plotting.
Plot	control parameters of the plot, see link{setMultiPlot} and **Details** for details.
yaxis.orient	orientation of the y-axis values, must be either "table" or "grid." "Table" is sorted from top to bottom, "grid" is sorted from bottom to top.
yaxis.order	the order of the y-axis values, must be one of "none," "ascending," or "descending."
yaxis.grid	logical, if TRUE, then draw grid lines.
xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis. See **Details**.
ylabels	set up y-axis labels.
xlabels	set up x-axis labels.
xtitle	x-axis title (also called x-axis caption).
ytitle	y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	arguments for specific methods.
jitter.y	logical, if TRUE, then adjust y values to reduce overlap for each group, or adjust randomly if no groups. If FALSE, then no adjustment is made.

Details

The what component of the Plot argument must be either "points" or "none."

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Methods

signature(x = "numeric")) Create a dot plot for numeric x-coordinate data and any (discrete) y-coordinate data.

signature(x = "Date")) Create a dot plot for Date x-coordinate data and any (discrete) y-coordinate data.

Note

A call should be made to setPage to set up the graphics environment before calling dotPlot.

See Also

setPage, boxPlot

Examples

```
## Not run:
set.seed(1)
X <- runif(12)
Y <- LETTERS[1:12]
setGD()
dotPlot(X, Y)
# For more details of dotPlot see
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

---

ecdfPlot *Empirical Distribution Plot*

Description

Creates a graph of the empirical distribution function of data.

Usage

```
ecdfPlot(x, group = NULL, Plot = list(name = "Auto", what =
  "stairstep", type = "solid", width = "standard", symbol = "circle",
  filled = TRUE, size = 0.09, color = "Auto"), xaxis.log = TRUE,
  xaxis.range = c(NA, NA), xlabels = 11, ylabels = 5,
  ytitle = "Cumulative Probability", xtitle = deparse(substitute(x)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## Default S3 method:
ecdfPlot(x, group = NULL, Plot = list(name = "Auto",
  what = "stairstep", type = "solid", width = "standard", symbol =
  "circle", filled = TRUE, size = 0.09, color = "Auto"),
  xaxis.log = TRUE, xaxis.range = c(NA, NA), xlabels = 11,
  ylabels = 5, ytitle = "Cumulative Probability",
  xtitle = deparse(substitute(x)), caption = "", margin = c(NA, NA,
  NA, NA), ...)
```

Arguments

x	the data to plot.
group	create groups for x. Each group is plotted as a separate line.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.

xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis. See **Details**.
xlabels	set the x-axis labels. See linearPretty for details.
ylabels	set the y-axis labels. See linearPretty for details.
ytitle	the y-axis title (also called y-axis caption).
xtitle	the x-axis title (also called x-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	any additional arguments needed by specific methods.

Details

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling ecdfPlot.

See Also

setPage, probPlot

Examples

```
## Not run:
set.seed(1)
X <- rlnorm(32)
setGD()
ecdfPlot(X)
# For more details of ecdfPlot see
vignette(topic="ProbabilityPlots", package="smwrGraphs")

## End(Not run)
```

getDist.fcn *Distribution Function*

Description

Gets the density, cumulative distribution, quantile, or random generation of the specified distribution. This function is used primarily as a support function for `probPlot`.

Usage

```
getDist.fcn(distribution, what = "q")
```

Arguments

`distribution`	the name of the distribution. See **Details**.
`what`	a character indicating which form to retrun. Must be "q" for quantile, "d" for density, "p" for cumulative distribution, or "r" for random generation.

Details

For general use, `distribution` should be one of "normal," "lognormal," "pearsonType3," "logpearsonType3," "exponential," "logistic," or "uniform." Partial matching is done, so only as many characters to make a unique match are required. Other distributions can be retrieved by specifying the base name of the distribution. That option can be useful if other packages that contain distribution functions have been loaded.

Value

The specified function.

See Also

`Normal`, `Lognormal`, `Exponential`, `Logistic`, `Uniform`, `PearsonIII`, `LogPearsonIII`,

histGram *Histogram*

Description

Creates either a frequency or density histogram.

Usage

```
histGram(x, breaks = "Sturges", Hist = list(), yaxis.range = c(NA,
  NA), ylabels = 7, xlabels = "Auto", xtitle = "", ytitle = "Auto",
  caption = "", margin = c(NA, NA, NA, NA), ...)

## Default S3 method:
histGram(x, breaks = "Sturges", Hist = list(type =
  "frequency", fill = FALSE, boundary = "lower", line.color = "black",
  fill.color = "gray80"), yaxis.range = c(NA, NA), ylabels = 7,
  xlabels = 7, xtitle = deparse(substitute(x)), ytitle = "Auto",
  caption = "", margin = c(NA, NA, NA, NA), ...)
```

Arguments

x	a numeric vector to create the histogram
breaks	any valid value for hist. See **Details**.
Hist	control parameters of the histogram. See **Details**.
yaxis.range	set the range for the y axis, the first value must be 0.
ylabels	the approximate number of labels for the y axis.
xlabels	the approximate number of labels for the x axis. The default value, "Auto" sets labels that are aligned with the breaks.
xtitle	x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption), "Frequency" for a frequency histogram, "Density" for a density histogram.
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	additional arguments for other methods.

Details

To set the x-axis range, you must specify numeric breaks that span the complete range of x.

The components of Hist:

type The type of the histogram. Must be one of "frequency" for actual counts in the bin, "density" for density in each bin, or "relative frequency" for percent in each bin.

fill Logical value, TRUE means each bin will be shaded with fill.color.

boundary Defines how values tied to bin limit boundaires are handled. If "upper," then the bin limit boundary is the upper limit of the range and values tied to that value are placed in the bin corresponding to the upper limit of the boundary. If "lower," then the bijn limit is the lower limit of the bin.

line.color The color of the lines around the bins.

fill.color The color the bins.

Value

Information about the graph.

Note

A call should be made to `setPage` to set up the graphics environment before calling `histGram`.

References

#'Helsel, D.R., and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water-Resources Investigations, book 4, chap. A3, 522 p.

See Also

`ecdfPlot`

Examples

```
## Not run:
set.seed(1)
Xbig <- rnorm(100)
setGD()
histGram(Xbig, breaks=seq(-3, 3, by=.5), Hist=list(type="density"))
# For more details of histGram see
vignette(topic="ProbabilityPlots", package="smwrGraphs")

## End(Not run)
```

---

hull *Construct a Hull*

Description

Constructs an enclosing hull from x- and y-coordinate data.

Usage

```
hull(x, y, percent = 100, smooth = FALSE)
```

Arguments

`x`	the x-coordinate data. Missing values are permitted, but ignored.
`y`	the y-coordinate data. Missing values are permitted, but ignored.
`percent`	the minimum percent to enclose.
`smooth`	logical, if `TRUE`, then smooth the bounding hull.

Value

A list containing the x- and y-coordinates of the hull.

See Also

dataEllipse, chull

Examples

```
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
# The enclosing polygon
hull(X, Y)
# The points
chull(X, Y)
## Not run:
# For examples of hull in graphs see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

---

interpLine *Interpolation*

Description

Creates a vector of interpolated points along a line drawn by one of the smwrGraphs functions.

Usage

```
interpLine(object, xfromy, yfromx, warn = TRUE, ...)
```

Arguments

object	an object created by one of the smwrGraphs functions.
xfromy	the y-axis coordinate values to use to create matching x-coordinate values. Missing values are permitted but result in missing values in the output.
yfromx	the x-axis coordinate values to use to create matching y-coordinate values. Missing values are permitted but result in missing values in the output.
warn	logical, if TRUE, then supress the warning message from probability or transformed axes plots.
...	any future additional arguments.

Details

Exactly one of xfromy or yfromx must be specified in the call.

Value

A vector of numeric values corresponding to those values in either `xfromy` or `yfromx`.

Note

The back-transformation information is not included in the output from the graphics functions. This primarily affects the `transPlot` function.

See Also

`transPlot`

Examples

```
## Not run:
# See for examples of interpLine:
demo(topic="FlowDur-Measurements", package="smwrGraphs")

## End(Not run)
```

`labelPoints` *Label Points*

Description

Labels points on a graph.

Usage

```
labelPoints(x, y, labels, dir = "E", offset = 0.75, size = 8,
  color = "black", current = list(yaxis.log = FALSE, yaxis.rev = FALSE,
  xaxis.log = FALSE))
```

Arguments

`x`	the x-axis data. Missing values are permitted, but ignored.
`y`	the y-axis data. Missing values are permitted, but ignored.
`labels`	the text labels, must be the same length as x and y. Missing values are permitted, but ignored.
`dir`	the direction relative to the point to place the label. See **Details**.
`offset`	the relative offset from the point. See **Details**.
`size`	character size in points.
`color`	the color of the labels.
`current`	the current plot controls. Typically, this would be the output from one of the graph creation functions like `xyPlot`.

Details

The value for `dir` can be of length one or the length of `x`, in which case, `dir` applies to the corresponding point. The value must be "N," "NE," "E," "SE," "S," "SW," "W," or "NW" cooresponding to the compass direction or "C" to center the label.

The value for `offset` can be of length one or the length of `x`, in in which case, `offset` applies to the corresponding point. The value is relative to the size of the test. The default value of `offset` is correct for creating a text plot, where the text is centered on the value.

Value

A list containing `x`, `y`, and `labels`.

Note

The current version does not have any method to automatically eliminate overlapping labels. Several iterations may be required trying various values for `dir` and possibly `offset` to produce non overlapping labels.

See Also

addAnnotation,, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
xyPlot(X, Y)
# Label the first point
labelPoints(X[1], Y[1], "First")
# For more details of labelPoints see
vignette(topic="GraphAdditions", package="smwrGraphs")

## End(Not run)
```

---

linearPretty *Pretty Axis*

Description

Constructs information for making a nicely formatted linear numeric axis. A support function for creating linear axes.

Usage

```
linearPretty(x, hard = FALSE, labels = "Auto", style = "Auto",
  extend.pct = 0, extend.range = TRUE)
```

Arguments

x	data defining the range to be plotted on the axis. Missing values are permitted, but ignored.
hard	logical, if TRUE, then use the minimum and maximum of x as the fixed range of the axis, otherwise find "nice" limits.
labels	either "Auto," which lets the function decide how many labels, the approximate number of labels, or the actual labels to use. If the actual labels are numeric, then the will be formatted using style. If they are character, then they must be able to be converted to numeric values (commas are removed before conversion).
style	a character string indicating the style of the axis labels if they are not specifically listed in labels. Valid values are "numeric," which forces the labels to be displayed as numbers; "scientific," which forces the labels displayed using scientific notation; or "Auto" (the default), which disaplays labels as numbers but switches to scientific notation for large ranges. Only the first letter is required. Any invalid value will produce simply formatted labels.
extend.pct	extend the axis range by extend.pct. Only valid when hard is FALSE.
extend.range	if TRUE, then extend the data range by a bit to avoid plotting on axis. Otherwise do not extend the data range. Only valid when hard is FALSE; Ignored in logPretty

Value

Information about the axis labels.

See Also

areaPlot, boxPlot, colorPlot, areaPlot, dotPlot, ecdfPlot, probPlot, qqPlot, scalePlot, splomPlot, timePlot, xyPlot

---

lineWt *Line Weights*

Description

Computes the weight, or width, of a line. Used primarily as a support function.

Usage

```
lineWt(x)

frameWt()

stdWt(x = 1)
```

Arguments

x	for lineWt, one of "standard" (0.7 pt), "color" (0.8 pt), "bold" (1.0 pt), or "hairline" (0.5 pt). For stdWt, a multiplier to set the line weight.

Value

The width of the line to set as the lwd graphics parameter.

See Also

par

---

logPretty *Pretty Axis*

Description

Constructs information for making a nicely formatted log-scale numeric axis. A support function for creating logarithmic axes.

Usage

```
logPretty(x, hard = FALSE, labels = "Auto", style = "Auto",
  extend.pct = 0, extend.range = NA)
```

Arguments

x	data defining the range to be plotted on the axis. Missing values are permitted, but ignored.
hard	logical, if TRUE, then use the minimum and maximum of x as the fixed range of the axis, otherwise find "nice" limits.
labels	either "Auto," which lets the function decide how many labels, the approximate number of labels, or the actual numeric values of the labels.
style	a character string indicating the style of the axis labels. Valid values are "numeric," which forces the labels to be displayed as numbers; "scientific," which forces the labels displayed using scientific notation; or "Auto" (the default), which disaplays labels as numbers but switches to scientific notation for large ranges. Only the first letter is required. Any invalid value will produce simply formatted labels.
extend.pct	extend the axis range by extend.pct. Only valid when hard is FALSE.
extend.range	required for naming consistency with other functions, not used.

Value

Information about the axis lables.

See Also

areaPlot, boxPlot, colorPlot, areaPlot, dotPlot, ecdfPlot, probPlot, qqPlot, scalePlot, splomPlot, timePlot, xyPlot

---

`month.USGS` *Month Abbreviations*

Description

A vector of USGS-style month abbreviations.

Usage

```
month.USGS
```

Format

A named character vector of the 12 preferred forms for month abbreviations.

References

Hansen, W.R., 1991, Suggestions to Authors of the United States Geological Survey, 7th Ed.: U.S. Govenement Printing Office, 289 p.

Examples

```
print(month.USGS)
## Not run:
# For examples of month.USGS in graphs see
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

`namePretty` *Pretty Axis*

Description

Constructs information for making a nicely formatted axis for discrete data. A support function for creating discrete axes.

Usage

```
namePretty(x, orientation = "table", order = "none",
  label.abbr = FALSE, offset = 0.5, style = "at")
```

Arguments

`x`	the discrete data values.
`orientation`	the orientation of the data in `x`.

"table"	first in sequence at top (ends on right if x-axis)
"grid"	first in sequence at bottom

`order`	the order of the data in `x`.

"none"	accept order as is
"ascending"	sort in ascending alphabetical order
"descending"	sort in descending alphabetical order
named numeric vector	sort by values (largest value at top if orientation is "table")
character vector	specifies the sequence of names

`label.abbr`	logical, if `TRUE`, then create abbreviations for `x`, otherwise use the full text of `x` for labels.
`offset`	amount to offset the range, generally 0.5 or 1. The range of the data is from 1 to the number of elements in `x`.
`style`	character string indicating the placement of the ticks. If "at" (default), then place ticks at the labels. If "between," then place ticks between the labels.

Value

Information about the axis labels

See Also

`dotPlot`

`numericData` *Numeric Values*

Description

Converts data to numeric values (support function).

Usage

```
numericData(x, lev = NULL)
```

Arguments

x	any vector that can be converted to numeric.
lev	levels for character data that represent categories rather than character representations of numeric values.

Value

Numeric data represented by x.

---

paraSpline *Parametric Spline*

Description

Constructs a parametric interpolating spline for x and y data. The x data are not required to be strictly increasing. Used as a support function.

Usage

```
paraSpline(x, y, n)
```

Arguments

x	the x-coordinate data. Missing values are not permitted.
y	the y-coordinate data. Missing values are not permitted.
n	The number of points in the output parametric spline fit.

Value

A list containing the components x and y, which are the coordinates of the parametric spline.

Examples

```
paraSpline(c(1,2,3), c(0,1,0), n=5)
```

piperPlot *Piper Diagram*

Description

Produces a Piper diagram.

Usage

```
piperPlot(xCat, yCat, zCat, xAn, yAn, zAn, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), axis.range = c(0, 100),
  num.labels = 6, ticks = FALSE, grids = !ticks,
  xCat.title = "Calcium", yCat.title = "Magnesium",
  zCat.title = "Sodium plus Potassium",
  xAn.title = "Chloride, Fluoride, Nitrite plus Nitrate",
  yAn.title = "Carbonate plus Bicarbonate", zAn.title = "Sulfate",
  x.yCat.title = "Calcium plus Magnesium",
  x.zAn.title = "Sulfate plus Chloride", units.title = "Percent",
  caption = "", margin = c(NA, NA, NA, NA))
```

Arguments

xCat	data for the cation x-axis, generally calcium.
yCat	data for the cation y-axis, generally magnesium.
zCat	data for the cation z-axis, generally sodium plus potasium.
xAn	data for the anion x-axis, generally chloride plus other minor constituents.
yAn	data for the anion y-axis, generally carbonate plus bicarbonate.
zAn	data for the anion z-axis, generally sulfate.
Plot	control parameters of the plot, see link{setMultiPlot} and **Details** for details.
axis.range	the range of the axes. Must be either c(0, 1) or c(0, 100).
num.labels	the number of labels to draw on each axis. Best selections are 2 giving (0, 100), 3 (0, 50, 100), 5 (0, 25, 50, 75, 100), or 6 (o, 20, 40, 60, 80, 100).
ticks	logical, if TRUE, then draw ticks.
grids	logical, if TRUE, then draw grid lines.
xCat.title	title (also called caption) for the cation x-axis.
yCat.title	title (also called caption) for the cation y-axis.
zCat.title	title (also called caption) for the cation x-axis.
xAn.title	title (also called caption) for the anion x-axis.
yAn.title	title (also called caption) for the anion y-axis.
zAn.title	title (also called caption) for the anion z-axis.
x.yCat.title	title for the cation x- and y-axis for the central Piper graph.

x.zAn.title	title for the anion x- and z-axis for the central Piper graph.
units.title	the units titles, should be either "Percent" of "Proportion" depending on axis.range.
caption	the figure caption.
margin	set up the plot area margins—ignored, included for consistency with other plotting functions in this package.

Details

The what component of the Plot argument must be either "points" or "none."

The units for xCat, yCat, zCat, xAn, yAn, and zAn should be in milli-equivalents.

Value

Information about the graph and current plot.

Note

A call should be made to setPage to set up the graphics environment before calling piperPlot.

References

Hem J.D., 1989, Study and interpretation of the chemical characteristics of natural water: U.S. Geological Survey Water-Supply Paper 2254, 263 p.

See Also

setPage, setMultiPlot, ternaryPlot, addPiper

Examples

```
## Not run:
# See for examples of piperPlot:
vignette(topic="PiperPlot", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")

## End(Not run)
```

---

piperSubplot *Piper Diagram*

Description

Plots the Piper diagram projected from the two trilinear diagrams on either side (support function).

Usage

```
piperSubplot(x, y, what = "points", symbol = rep(1, length(x)),
  color = rep(1, length(x)), size = rep(0.05, length(x)),
  axis.range = c(0, 100), num.labels = 6, ticks = FALSE,
  grids = !ticks, x1title = "x1", y1title = "y1", x2title = "x2",
  y2title = "y2", plot = TRUE)
```

Arguments

x	x-axis coordinate values (derived from the cation z-axis).
y	y-axis coordinate values (derived from the anion y-axis).
what	the type of plot, must be either "points" or "lines."
symbol	the symbol to use if what is "points."
color	the color of the plot.
size	the size of the symbol if what is "points."
axis.range	the range of the axes. Must be either c(0, 1) or c(0, 100).
num.labels	the number of labels to draw on each axis.
ticks	logical, if TRUE, then draw ticks.
grids	logical, if TRUE, then draw grid lines.
x1title	the title (also called caption) of the bottom x-axis.
y1title	the title (also called caption) of the left y-axis.
x2title	the title (also called caption) of the top x-axis.
y2title	the title (also called caption) of the right y-axis.
plot	logical, if TRUE, then plot the data.

Details

Support function, to be called only from piperPlot.

Value

If plot is TRUE, then the range of the user coordinates. Otherwise, the transformed x- and y-coordinate values.

See Also

piperPlot

preSurface *Prepare for surface plot*

Description

Selects the projection for a surface plot.

Usage

```
preSurface(x, y, z.surf, zaxis.log = FALSE, zaxis.range = c(NA, NA),
  yaxis.log = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = c(NA, NA), xlabels = "Auto", ylabels = "Auto",
  zlabels = "Auto", phi = NA, theta = NA, batch = FALSE)
```

Arguments

x	the x-axis coordinate data, must be strictyly increasing. Missing values not permitted. May be of type Date.
y	the y-axis coordinate data, must be strictyly increasing. Missing values not permitted.
z.surf	a numeric matrix representing the surface. The length of x must match the number of rows in z.surf. The length of y must match the number of columns in z.surf.
zaxis.log	logical, if TRUE, then log-transform the z axis.
zaxis.range	set the range of the z-axis.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.range	set the range of the y-axis.
xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis.
xlabels	set up x-axis labels.
ylabels	set up y-axis labels.
zlabels	set up z-axis labels.
phi	the viewing angle relative to the x-y plane. If NA, then programmatically select a reasonable angle. Should be greater than 0 and less than 90 degrees, but the best angles are generally between 20 and 45 degrees.
theta	the viewing angle relative to the x-axis. Positive values rotate the x-y plane in a clock-wise direction. If NA, then programmatically select a reasonable angle by putting the largest values of z.surf near the rear and the smallest values near the front. In general, the best angles are not multiples of 90 degrees.
batch	logical or character. If logical and TRUE, then select the viewing angles specified by phi and theta. If logical and FALSE, then draw 9 candidate combinations of viewing angles and pause for user input to select the desired viewing angles. If character, then select the viewing angles specified by the letter selection, must be A through I.

Value

A list containg the projection information and the data for plotting. Must be used in the call to `surfacePlot`

See Also

`surfacePlot`

Examples

```
## Not run:
# See for examples of preSurface:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

print.Layout *Print Objects*

Description

Prints the layout of a figure.

Usage

```
## S3 method for class 'Layout'
print(x, ...)
```

Arguments

`x`	the object to print
`...`	not used, required for other methods

Value

The object `x` is returned invisibly.

Note

The layout of graphs is displayed.

See Also

`setLayout`, `setGraph`

probPlot *Probability Plot*

Description

Creates a probability plot.

Usage

```
probPlot(x, truncate = NA, FLIP = FALSE, distribution = "normal",
  alpha = 0.4, Plot = list(name = "", what = "points", type = "solid",
  width = "standard", symbol = "circle", filled = TRUE, size = 0.09, color
  = "black"), yaxis.log = TRUE, yaxis.range = c(NA, NA),
  ylabels = 11, xlabels = 11, CDF = !RI, xtitle = ifelse(CDF,
  "Cumulative Probability", "Exceedence Probability"), RI = FALSE,
  RItitle = "Recurrence Interval, in years",
  ytitle = deparse(substitute(x)), caption = "", margin = c(NA, NA,
  NA, NA), ...)

## Default S3 method:
probPlot(x, truncate = NA, FLIP = FALSE,
  distribution = "normal", alpha = 0.4, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = TRUE,
  yaxis.range = c(NA, NA), ylabels = "Auto", xlabels = 11,
  CDF = !RI, xtitle = ifelse(CDF, "Cumulative Probability",
  "Exceedence Probability"), RI = FALSE,
  RItitle = "Recurrence Interval, in years",
  ytitle = deparse(substitute(x)), caption = "", margin = c(NA, NA,
  NA, NA), ...)
```

Arguments

x	the data to plot. Missing values are allowed and ignored.
truncate	truncate the data at the specified value. See **Details**.
FLIP	if TRUE, then plot the cumulative distribution. Otherwise, plot as flipped data (largest values on left).
distribution	the name of the desired function converting from probabilities to coordinates.
alpha	the alpha value of the function for computing plotting positions.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.range	set the range of the y axis. See **Details**.
ylabels	set the y-axis labels. See logPretty for yaxis.log set to TRUE or linearPretty for yaxis.log set to \ceodeFALSE for details.

xlabels	set the x-axis labels. See probPretty for details.
CDF	logical, if TRUE, then label with increasing probabilities. Otherwise label with decreasing probabilities.
xtitle	the x-axis title (also called x-axis caption).
RI	logical, if TRUE, then label the top axis with recurrence intervals. If RI is set to TRUE, then CDF will be set to FALSE.
RItitle	the top x-axis title if RI is TRUE.
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	parameters for the distribution function. If any parameter is specified, then an attempt is made to draw the fit between the computed distribution and the observed data.

Details

Truncation of the data to plot (x) results in a conditional probability plot. For any numeric value for truncate, the values in x less than or equal to truncate are not plotted and the remaining values are plotted at their conditional probability (the probability computed with all values). The behavior for the default value for truncate = NA, depends on yaxis.log. If yaxis.log is TRUE, then truncate is treated as though it was 0; otherwise truncate is treated as though it was -Inf, which results in no truncation.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling probPlot.

See Also

setPage, ecdfPlot, qqPlot

Examples

```
## Not run:
set.seed(1)
X <- rlnorm(32)
setGD()
probPlot(X)
```

```
# For more details of probPlot see
vignette(topic="ProbabilityPlots", package="smwrGraphs")
demo(topic="FlowDur-Measurements", package="smwrGraphs")

## End(Not run)
```

---

probPretty *Pretty Axis*

Description

Constructs information for making a nicely formatted probability axis. A support function for creating probability axes.

Usage

```
probPretty(x, hard = FALSE, labels = "Auto", style = "probability",
  exceedence = TRUE, priority = "label", distribution = "normal",
  ...)
```

Arguments

x	axis coordinates in range 0-1 or 0-100 allowed and assumed if max(x) > 1. Note that only the range (min and max) are needed. Missing values allowed, but ignored.
hard	logical force min(x) and max(x) as axis limits, otherwise use "nice" limits.
labels	an estimate of the number of labels desired, or specific label points. If vector, then can be expressed as character strings, which are converted to numeric and automatically scaled. Default is "Auto", which is 9 if minimum x is greater than .01 and 11 otherwise.
style	can be either "probability" or "percent" indicates how the labels are formatted.

TRUE	exceedence probs and additional recurrence interval labels
FALSE	cumulative probabilities

exceedence

"label"	"nice" labels given priority for selection
"positions"	uniform separation given priority for selection

priority

distribution	the name of the probability function, defaults to normal.
...	options for the distribution function.

Value

Information about the axis labels.

See Also

probPlot

---

qqPlot *Q-Q Plot*

Description

Creates a quantile-quantile (q-q) or a q-normal plot.

Usage

```
qqPlot(x, y, alpha = 0.4, Plot = list(name = "Paired data quantiles",
  what = "points", type = "solid", width = "standard", symbol = "circle",
  filled = TRUE, size = 0.09, color = "black"), LineRef = list(name =
  "Line of best fit", what = "lines", color = "black"),
  Line1.1 = list(name = "Line of equality", what = "lines", color =
  "gray"), yaxis.log = FALSE, yaxis.range = c(NA, NA),
  xaxis.log = FALSE, xaxis.range = c(NA, NA), ylabels = 7,
  xlabels = 7, xtitle, ytitle, caption = "", margin = c(NA, NA, NA,
  NA), ...)

## Default S3 method:
qqPlot(x, y, alpha = 0.4, Plot = list(name =
  "Paired data quantiles", what = "points", type = "solid", width =
  "standard", symbol = "circle", filled = TRUE, size = 0.09, color =
  "black"), LineRef = list(name = "Line of best fit", what = "lines",
  color = "black"), Line1.1 = list(name = "Line of equality", what =
  "lines", color = "gray"), yaxis.log = FALSE, yaxis.range = c(NA, NA),
  xaxis.log = FALSE, xaxis.range = c(NA, NA), ylabels = 7,
  xlabels = 7, xtitle, ytitle, caption = "", margin = c(NA, NA, NA,
  NA), ...)
```

Arguments

x	the x-axis data, or data to plot if y is missing.
y	the y-axis data. If missing, then produce a quantile-normal quantile plot from the data in x.
alpha	the alpha value of the function for computing plotting positions.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
LineRef	control parameters of the reference line (best fit between x and y. See **Details**.

Line1.1	control parameters for the 1:1 line. Drawn only for q-q plot. See **Details**.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.range	set the range of the y axis. See **Details**.
xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis. See **Details**.
ylabels	set the y-axis labels. See linearPretty for details.
xlabels	set the x-axis labels. See linearPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	any additional arguments required for specific methods.

Details

The argument what for either LineRef or Line1.1 may be set to "none" to suppress drawing of either line.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling qqPlot.

See Also

setPage, ecdfPlot, probPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
setGD()
qqPlot(X)
# For more details of qqPlot see
vignette(topic="ProbabilityPlots", package="smwrGraphs")

## End(Not run)
```

refLine *Reference Line*

Description

Adds a reference line (vertical, horizontal, or regression) to a graph.

Usage

```
refLine(horizontal, vertical, coefficients, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), current = list(yaxis.log = FALSE,
  yaxis.rev = FALSE, xaxis.log = FALSE), xrange = c(NA, NA),
  yrange = c(NA, NA), log10 = FALSE)
```

Arguments

horizontal	draw horizontal lines at the specified values.
vertical	draw vertical lines at the specified values.
coefficients	draw a fitted line from the coefficients of a regression model.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters. The argument what is forced to "lines."
current	the current parameters of the graph. Typically, this would be the output from one of the graph creation functions like xyPlot.
xrange	limit x-axis range for horizontal or regression lines
yrange	limit y-axis range for vertical lines
log10	logical, if TRUE, then log base 10 transform used in the regression model, otherwise either the natural log was used or no transform.

Value

Information about the graph.

See Also

addXY, addSmooth, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
xyPlot(X, Y)
# Add the 1:1 line
```

```
refLine(coefficient=c(0,1))
# For more details of refLine see
vignette(topic="GraphAdditions", package="smwrGraphs")
demo(topic="Coplot-complexScatterPlot", package="smwrGraphs")

## End(Not run)
```

`renderBoxPlot` *Box Plot*

Description

Draws a box plot (support functions).

Usage

```
renderBoxPlot(xtoplot, stats, Box, explan, expz, yaxis.log = FALSE,
  yrange = c(NA, NA), xrange = range(xtoplot) + c(-1, 1),
  ylabels = "Auto", xlabels = "Auto", xlabels.rotate = FALSE,
  xtitle = "", ytitle = "", caption = "", margin = c(NA, NA, NA,
  NA))

renderBXP(x, width, z, draw.RL = TRUE, fill = "none")
```

Arguments

`xtoplot`	the x-axis locations for each boxplot.
`stats`	a list containing the statistics for the boxplots.
`Box`	a list containing the control info for the boxplots.
`explan`	a list containing the information for an explanation.
`expz`	a list containing the information for an explanation of the boxplot.
`yaxis.log`	logical, if `TRUE`, then use a log transform for the data and the y-axis.
`yrange`	set the y-axis range.
`xrange`	set the x-axis range.
`ylabels`	either "Auto," the approximate number of labels, or the actual labels to use for the y-axis.
`xlabels`	either "Auto" or the x-axis labels for each boxplot.
`xlabels.rotate`	logical, if `TRUE`, then rotate the x-axis labels by 90 degrees.
`xtitle`	the x-axis title (also called x-axis caption).
`ytitle`	the y-axis title (also called y-axis caption).
`caption`	the figure caption.
`margin`	the parameters of the margin of the plot area.
`x`	the x-coordinate for the box.

width	the width of the box in x-axis units.
z	a list containing the statistics for the individual boxplot.
draw.RL	logical, if TRUE, then draw the reporting level for the individual boxplot.
fill	a character string describing the fill color for the box or "none" for no fill.

Value

Information about the graph.

Note

The function renderBXP draws a single boxplot in a graph. The function renderBoxPlot is called from each method function for boxPlot to produce the boxplot.

See Also

boxPlot

---

renderPretty *Label Axes*

Description

Draws ticks, labels, or grids for an axis (support functions).

Usage

```
ticks.render(arg1, side, lwd)

renderY(pretty, left = list(ticks = TRUE, labels = TRUE, grid = FALSE,
  finegrid = FALSE), right = list(ticks = TRUE, labels = FALSE, grid =
  FALSE, finegrid = FALSE), lefttitle = "Y-AXIS TITLE",
  righttitle = "")

renderX(pretty, bottom = list(ticks = TRUE, labels = TRUE, grid = FALSE,
  finegrid = FALSE, angle = 0), top = list(ticks = TRUE, labels = FALSE,
  grid = FALSE, finegrid = FALSE, angle = 0), bottitle = "X-AXIS TITLE",
  toptitle = "", caption = "")
```

Arguments

arg1	control parameters for the tick locations.
side	the number of the axis, 1 is bottom, 2 is left, and so forth.
lwd	the line weight for the ticks.
pretty	the output from one of the "pretty" functions.
left	control parameters for the left y axis.

right	control parameters for the right y axis.
lefttitle	the title for the left y axis.
righttitle	the title for the right y axis.
bottom	control parameters for the bottom x axis.
top	control parameters for the top x axis.
bottitle	the title for the bottom x axis.
toptitle	the title for the top x axis.
caption	the figure caption.

Value

Nothing is returned.

`reportGraph` *Report Graph*

Description

Creates a report of any R object in a graph.

Usage

```
reportGraph(x, family = "Auto", size = 60 * par("csi"))
```

Arguments

x	any R object.
family	the font family to use in the report. See **Details**.
size	the size of the text, in points

Details

The value for `family` can be any valid font family for the device. In general "serif," "sans," "mono," and "USGS" are valid. The default, "Auto" selects "USGS" for character vectors, and "mono" for any other object.

Value

In contrast to other high-level graphics functions in the `smwrGraphs` package, this function returns nothing because nothing is expected to be added to the graph and nothing contributed to a possible explanation.

Note

The report is always placed in the upper left hand corner of the graph and is left justified. If the report is longer than the height of the graph or wider than the width of the graph, then the report is truncated.

Examples

```
## Not run:
setGD()
reportGraph("Hello world!")
# For more details of reportGraph see
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

`scalePlot` *Scale Plot*

Description

Produces a graph with a fixed aspect ratio for the x- and y-axes.

Usage

```
scalePlot(x, y, scale = 1, Plot = list(name = "", what = "lines", type
  = "solid", width = "standard", symbol = "circle", filled = TRUE, size =
  0.09, color = "black"), yaxis.log = FALSE, yaxis.rev = FALSE,
  yaxis.range = c(NA, NA), xaxis.log = FALSE, xaxis.range = c(NA,
  NA), ylabels = 7, xlabels = 7, xtitle = deparse(substitute(x)),
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA))
```

Arguments

`x`	the x-axis data.
`y`	the y-axis data.
`scale`	the y/x ratio. See **Details**.
`Plot`	parameters defining the characteristics of the plot. See `setPlot` for a description of the parameters.
`yaxis.log`	logical, if `TRUE`, then log-transform the y axis.
`yaxis.rev`	logical, if `TRUE`, then reverse the y axis.
`yaxis.range`	set the range of the y axis. See **Details**.
`xaxis.log`	logical, if `TRUE`, then log-transform the x axis.
`xaxis.range`	set the range of the x axis. See **Details**.
`ylabels`	set the y-axis labels. See `linearPretty` for details.
`xlabels`	set the y-axis labels. See `linearPretty` for details.
`xtitle`	the x-axis title (also called x-axis caption).
`ytitle`	the y-axis title (also called y-axis caption).
`caption`	the figure caption.
`margin`	set the plot area margins, in units of lines of text. Generally all NA or the output from `setGraph` if appropriate.

Details

The scale argument sets the scaling ratio of the y-axis to the x-axis. For latitude and longitude data, set the scale to 1/cos(midlat/180*pi), where midlat is the midrange of the latitude.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling scalePlot.

See Also

setPage, xyPlot

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
scalePlot(X, Y, Plot=list(what="points", size=0.05))
# For more details of scalePlot see
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

---

seasonPlot *Season Plot*

Description

Creates a plot of data on a yearly cycle.

Usage

```
seasonPlot(x, y, Plot = list(), yaxis.log = FALSE, yaxis.rev = FALSE,
  yaxis.range = c(NA, NA), xaxis.range = "", ylabels = 7,
  xlabels = 7, xtitle = "", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'ANY,numeric'
```

```
seasonPlot(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA),
  xaxis.range = c("calendar", "water", "climate"), ylabels = 7,
  xlabels = "Auto", xtitle = "", ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'character,numeric'
seasonPlot(x, y, Plot = list(name = "",
  what = "lines", type = "solid", width = "standard", symbol = "circle",
  filled = TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA),
  xaxis.range = c("calendar", "water", "climate"), ylabels = 7,
  xlabels = "Auto", xtitle = "", ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)
```

Arguments

x	the x-coordinate data.
y	the y-coordinate data.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.rev	logical, if TRUE, then reverse the y axis.
yaxis.range	set the range of the y-axis.
xaxis.range	set the range of the x-axis. Must be one of "calendar," "water," or "climate" to set the type of year that is shown on the x-axis.
ylabels	set up y-axis labels. See linearPretty for details.
xlabels	set up x-axis labels. See **Details**.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	arguments for specific methods.

Details

For seasonPlot, the value for xlabels must be one of "full," the full month names; "abbrev," abbreviations; or "letter," the first letter of the month. The default is "Auto," which will select an appropriate labeling scheme.

Value

Information about the graph.

Methods

signature(x = "ANY", y = "numeric") Create a seasonal plot for any valid date data and numeric y data.

signature(x = "character", y = "numeric") Create a seasonal plot for date data in the form of month and day, like "Jan 01" or "Jaunary 01." Typically used to plot daily mean values.

Note

A call should be made to `setPage` to set up the graphics environment before calling `seasonPlot`.

To add a plot to the graph created by `seasonPlot`, the x-axis data must be expressed as decimal time relative to January 1. The function `baseDay2decimal` can be used to convert data in the form of base day to decimal time.

See Also

`setPage`, `timePlot`, `baseDay2decimal`

Examples

```
## Not run:
# the months function is in lubridate
X <- as.Date("2001-01-15") + months(0:11)
set.seed(1)
Y <- runif(12)
setGD()
seasonPlot(X, Y)
# For more details of seasonPlot see
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

---

seriesPlot *Series Plot*

Description

Creates a plot of a regular series on a seasonal cycle; the annual values for each season are plotted.

Usage

```
seriesPlot(x, SeasonLine = list(name = "", what = "vertical", color =
  "black"), SeasonPoint = list(name = "", what = "points", symbol =
  "circle", filled = TRUE, size = 0.09, color = "black"),
  yaxis.log = FALSE, yaxis.range = c(NA, NA), ylabels = 7, xlabels,
  xtitle = "", ytitle = "", caption = "", margin = c(NA, NA, NA,
  NA), ...)
```

```
## Default S3 method:
seriesPlot(x, SeasonLine = list(name = "", what =
  "vertical", color = "black"), SeasonPoint = list(name = "", what =
  "points", symbol = "circle", filled = TRUE, size = 0.09, color =
  "black"), yaxis.log = FALSE, yaxis.range = c(NA, NA), ylabels = 7,
  xlabels = frequency(x), xtitle = "",
  ytitle = deparse(substitute(x)), caption = "", margin = c(NA, NA,
  NA, NA), ...)
```

Arguments

x	data that can be treated as a regularly-spaced time series. Missing values are permitted, but result in missing seasons.
SeasonLine	control parameters of the lines in the plot. See **Details**.
SeasonPoint	control parameters of the points in the plot. See **Details**.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.range	set the range of the y axis. See **Details**.
ylabels	set the y-axis labels. See linearPretty for details.
xlabels	set the x-axis labels and number of seasons when x is a simple numeric vector, may be a single numeric value indicating the number of seasons in x or a vector of the names of the seasons. See namePretty for details. If X is a time-series object, then the labels are set to the frequency characteristic of x.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	any additional arguments required for specific methods.

Details

The argument what for SeasonLine must be either "lines" or "vertical." See monthplot for more information.

The argument what for SeasonPoint can be set to "none" to suppress drawing of symbols or "points" to draw symbols at the ends of the line segments described by SeasonLine.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Note

A call should be made to `setPage` to set up the graphics environment before calling `seriesPlot`.

See Also

`setPage`, `seasonPlot`, `monthplot`

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
setGD()
seriesPlot(X, xlabels=c("A", "B", "C", "D"))
# For more details of  seriesPlot see
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

---

setAxis *Set up an Axis*

Description

Sets up axis information (support function).

Usage

```
setAxis(data, axis.range, axis.log, axis.rev, axis.labels, ...)
```

Arguments

`data`	the coordinates for the particular axis
`axis.range`	set axis range.
`axis.log`	logical, if `TRUE`, then log transform the axis.
`axis.rev`	logical, if `TRUE`, then reverse the axis direction.
`axis.labels`	set axis labels.
`...`	additional arguments to the "pretty" functions.

Value

Information about the axis

See Also

`linearPretty`, `logPretty`

setColor *Colors*

Description

Checks or converts any data to valid colors (support function).

Usage

```
setColor(Color)
```

Arguments

Color	any kind of data that might be interpreted as a color.

Value

The values in Color converted to a value that could be interpreted as a color.

See Also

colors, rainbow

Examples

```
## Not run:
# See for examples of setColor:
vignette(topic="PiperPlot", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")

## End(Not run)
```

setDefaults *Default Values*

Description

Sets the default values for plot control lists (support function).

Usage

```
setDefaults(current = list(), ...)
```

Arguments

current	the control parameters specified in the call to the high-level graphing function.
...	the default values for each name required for the control parameters.

Value

The control parameters with defaults substituted for missing names.

`setExplan` *Explanation*

Description

Adds the current plot information to plot control list (support function).

Usage

```
setExplan(current, old = NULL)
```

Arguments

`current`	the current plot information.
`old`	existing explanation information.

Value

A list having four components:

`text` a list having two components:

text	the description in the explanation, derived from the `name` component in `Plot`
cex	the size of the `text` to write in the explanation

`lines` a list having seven components and controlling both lines and points:

type	the type of plot (specifying points, lines, and so forth)
lwd	the line weight
lty	the line type
pch	the symbol
csi	the size of the symbol in inches
cex	the the size of the symbol relative to character size
col	the color of the plot

`areas` a list having two components:

fill	the color of the fill area
borders	the color of the border

`current` a list like `current` with the defaults set

Each entry in `text` must have a corresponding entry in `lines` and `areas`.

See Also

`setPlot`

`setGD` *Graphics Page*

Description

Sets up a graphics page. The functions `setPage`, `setRStudio`, and `setGD` set up onscreen devices. The functions `setPDF`, `setSweave`, `setKnitr`, and `setPNG` set up files for graphics output.

Usage

```
setGD(name = "USGS")

setKnitr(name, width, height, ...)

setPDF(layout = "portrait", basename = "USGS", multiplefiles = FALSE)

setPNG(name, width, height, ...)

setPage(layout = "portrait", font = "preview", name = "USGS",
  multiple = FALSE, device = "default")

setRStudio()

setSweave(name, width, height, ...)
```

Arguments

`name`	the name of the graphics page or the filename for `setSweave`.
`width`	the width of the graphics page.
`height`	the height of the graphics page.
`...`	additional arguments, which are ignored by `setSweave`, `setKnitr`, and `setPNG`.
`layout`	A description of the orientation and shape of the graphics page. See **Details**.
`basename`	the base name of the pdf file name.

multiplefiles	logical, if TRUE, then modify basename to create multiple files for multiple pages.
font	a description of the font. The choices are "preview," which is 12 point Arial Narrow; "USGS," which is 8 point Arial Narrow; "EST," which is 8 point Times New Roman; "PPT," which is 24 point Arial; and "PDF," which is 8 point Arial Helvetica-Narrow. "PDF" should be chosen if the graphs are to be saved to a portable document format (pdf) file.
multiple	logical, if TRUE, then allow multiple pages.
device	the name of the graphics device. See **Details**.

Details

If layout is "portrait," then the page size is 8.5 by 11 inches and the graph area is 7.25 by 9.5 inches.

If layout is "landscape," then the page size is 11 by 8.5 inches and the graph area is 9.5 by 7.25 inches.

If layout is "square," then the page size is 7 by 7 inches and the graph area is 6.5 by 6.5 inches (setPage only).

If layout is "slide," then the page size is 10 by 7.5 inches and the graph area is 9.5 by 7.0 inches (setPage only).

Layout may also be a tagged list, with components width and height giving the width and height of the page, the width and height of the graph area is 0.5 inch less that the page, except for setPDF where it is 0.1 inch less.

The user may specify a graphics device other than the default for the system. This may be necessary when running under certain user environments like RStudio (TM).

Value

For setPage and setPDF, a list with two components: dev, the device number; and name, the name or basename. For setGD setSweave, setKnitr, setPNG, and setRStudio nothing is returned.

Note

The focus of all of the graphics functions is on producing near-publication ready figures for U.S. Geological Survey (USGS) report series. The function setPDF should be used to create the PDF files for those figures. The fonts that are used in graphs created by calling setPDF closely mimic those required in USGS reports. One peculiarity of the fonts is that bold expressions do not appear bold in the PDF, but the font is tagged bold.

The functions setSweave, setKnitr, and setPNG are graphics set up functions to be used when using Sweave, knitr and markdown, respectively. The functions setSweave and setPDF require a call to dev.off to close the graphics device after all graphics are completed; knitr and markdown automatically close the graphics device, so the call to dev.off is not needed in those scripts.

The function setRStudio is designed to set up the default graphics device in RStudio rather than open a separate graphics screen. This is useful for preview only as some features of the graphics system cannot be replicated on that graphics device.

The function setGD is designed to be a quick and easy graphics page setup function. It is designed to be used by functions to set up the graphics environment if the user fails to do so.

See Also

setLayout, setGraph

Examples

```
## Not run:
# See for examples of setGD:
demo(topic="AnnualFlowBarChart", package="smwrGraphs")
demo(topic="Coplot-complexScatterPlot", package="smwrGraphs")
demo(topic="Coplot-simpleBoxPlot", package="smwrGraphs")
demo(topic="DurationHydrograph", package="smwrGraphs")
demo(topic="FlowDur-Measurements", package="smwrGraphs")
demo(topic="HydroPrecip", package="smwrGraphs")
# See for examples of setPage:
demo(topic="PiperScript", package="smwrGraphs")
# See for examples of setPDF:
demo(topic="MeasurementRating", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")
demo(topic="RightAxisExample", package="smwrGraphs")
demo(topic="TopAxisExample", package="smwrGraphs")
# See for examples of setSweave:
vignette(topic="BoxPlots", package="smwrGraphs")
vignette(topic="DateAxisFormats", package="smwrGraphs")
vignette(topic="GraphAdditions", package="smwrGraphs")
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="LineScatter", package="smwrGraphs")
vignette(topic="PiperPlot", package="smwrGraphs")
vignette(topic="ProbabilityPlots", package="smwrGraphs")

## End(Not run)
```

---

setGraph *Set Graph*

Description

Sets up a specific graph on a graphics device.

Usage

```
setGraph(graphNum, layout, noTicks = NULL)
```

Arguments

graphNum	the number of the graph.
layout	the layout of the graph.
noTicks	suppress ticks on a specific axis.

Value

The parameters of the margin of the plot area.

Note

This function is called using the information generated by `setLayout`. It invisibly sets up the graphics device for the next graph.

See Also

`setLayout`

Examples

```
## Not run:
# See for examples of setGraph:
vignette(topic="BoxPlots", package="smwrGraphs")
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="PiperPlot", package="smwrGraphs")
demo(topic="HydroPrecip", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")

## End(Not run)
```

---

setGroupPlot *Plot Parameters*

Description

Sets plot control list for groups of data (support function).

Usage

```
setGroupPlot(current, Grps = 1, name = "", what = "points",
  type = "solid", width = "standard", symbol = "circle",
  filled = TRUE, size = 0.09, color = "black")
```

Arguments

`current`	the plot parameters specified in the call to the high-level graphing function.
`Grps`	the number of groups.
`name`	the name associated with the group. See **Details**.
`what`	what kind of plot. Must be one of "points," symbols only; "lines," line segments connecting points only; "both," line segments connecting isolated symbols; "overlaid," line segments connecting points with symbols; "stairstep," stairstep line segments; or "vertical," vertical lines from the y-axis origin to the y value at each x value.

type	the type of line, if drawn. Must be one of "solid," "dashed," "dotted."
width	the width of line, if drawn. Must be one of "standard," resulting in a line width of about 0.7 points; "color," resulting in a line width of about 0.8 points; "bold," resulting in a line width of about 1. point; or "hairline" resulting in a line width of about 0.5 points. Note these values are doubled if the font argument to setPage is "PPT."
symbol	type symbol, if drawn. Must be one of "circle," "uptri" (upward pointing triangle), "plus," "x," "diamond," "downtri" (downward pointing triangle), "square," or "dot."
filled	logical, if TRUE, then fill the symbol. Valid only for symbol equal to "circle," "uptri," "diamond," "downtri," or "square."
size	the size of the symol in inches, if drawn.
color	the color of the plotted values for each group. Can be a named color, such as "black" or "gray50" or an RGB color like "#A09623."

Details

If the name component in the original call to the high-level plot is "Auto," then the description for the explanation is taken from the value in the Group argument in that call. Otherwise the user must specify a name for each group.

If the color component in the original call to the high-level plot is "Auto," then the colors for each group are based on a sequence of 15 colors that are easily distinguished from each other. If there are more than 15 groups, then a gray scale is used with no guarantee of easily distinguished colors.

Value

A list having two components:

current	a list like current with the defaults set
Explan	a list for creating an explanation

See Also

setExplan, for details about the list required for an explanation.

---

setLayout *Graph Layout*

Description

Set up the layout of one or more graphs on a page.

Usage

```
setLayout(width = NULL, height = NULL, num.cols = max(1,
  length(width)), num.rows = max(1, length(height)),
  num.graphs = num.rows * num.cols, explanation = NULL,
  shared.x = -1, shared.y = -1, yleft = 3.5, yright = NA,
  xbottom = 3.2, xtop = NA)
```

Arguments

`width`	the width of the graph area, exclusive of any explanation. Can be either the total width or the width of each column of graphs. If NULL, then use default figure width. See **Details**.
`height`	the height of the graph area, exclusive of explanation. Can be either the total width or the height of each row of graphs. If NULL, then use defualt figure height.
`num.cols`	the number of columns in the rectangular array of graphs. Computed from width if not supplied.
`num.rows`	the number of rows in the rectangular array of graphs. Computed from height if not supplied.
`num.graphs`	the number of actual graphs in the rectangular array of graphs. Computed from `num.cols` and `num.rows` if not supplied. Note that if the EXPLANATION is to be placed in one of the array of graphs, then `num.graphs` must be less than the product of num.cols and num.rows.
`explanation`	a description of where to place the explanation if put into a separate graph. See **Details**.
`shared.x`	indicate how the x axes are to be shared. See **Details**.
`shared.y`	indicate how the y axes are to be shared. If < 0, then no sharing–each has own axis labels, etc. If = 0, then axes in direct contact. If > 0, then the value indicates the relative spacing.
`yleft`	space to allocate on the left margin of each graph.
`yright`	space to allocate on the right margin of each graph. See **Details**.
`xbottom`	space to allocate on the bottom margin of each graph.
`xtop`	space to allocate on the top margin of each graph. See **Details**.

Details

The layout of multple graphs on a page is always set up as a rectangular grid. The columns can be specified in one of two ways, either by specifying the width of each column using `width` or setting equal-width columns by specifying `num.cols`. The rows can be specified in one of two ways, either by specifying the height of each column using `height` or setting equal-height rows by specifying `num.rows`.

If an explanation is to be placed outside of the graphs, then `explanation` is used to indicate where the explanation is to be placed. The explanation can be placed either to the right of the grid of graphs, at the bottom of the grid, or in one of the grid cells.

To place an explanation to the right of the graphs, `explantion` should be set to `list(right=ewid)`, where `ewid` is the width of the explanation. In this case, the total of `width` and `ewid` must be less that the total available for the page.

To place an explanation at the bottom of the graphs, `explantion` should be set to `list(bottom=ehei)`, where `ehei` is the height of the explanation. In this case, the total pf `height` and `ehei` must be less than the total available for the page.

To place an explanation within a cell of the grid, `explantion` should be set to `list(grid=enum)`, where `enum` is the cell number in the grid. Cell numbers are sequential starting in the upper left and increasing by column. In this case `num.graphs` must be set to some number less than `num.cols` times `num.rows`.

The width of the explanation can be estimated by allocating 1 inch per 13 characters for `font` set to "preview" or 1 inch per 17 characters for `font` set to "USGS" plus 0.5 inch for the symbols. The width for box plots should be 2 inches for any type other that "tukey" and 2.5 inches for "tukey." The height of the explanation can be estimated as 1 inch per 8 lines of explanation for `font` set to "preview" and 1 inch per 10 lines of explanation for `font` set to "USGS"—allocate an extra 2 lines for the title. Box plots require about 3 inches for the truncated and simple types and about 4.5 inches for type "tukey" and about 4 inches for type "extended." The plot area within each cell is set up to have consistent widths within each column and consistent heights within each row.

The arguments `yleft`, `yright`, `xbottom`, and `xtop` are used to set up the plot area margins. If axes are not shared, then the margin values are set for any graph using those values. If the axes are shared, then the margin values apply to the corresponding left column, right column, bottom row or top row. The values for `yleft` and `xbottom` are useful defaults. If the y-axis labels are wider than typical values, such as those for very large numbers or names, then the value for `yleft` should be increased. If the x-axis labels are rotated, then the value for `xbottom` should be increased.

The arguments `shared.x` and `shared.y` control axis sharing. If the values are negative, then the axes are not shared and the margins are set as described in the preceding paragraph. If the values are nonnegative, then the axes are shared and the margin is set by the value. For example, a value of 0 means the axes are touching. A value of 1 generally gives enough spacing between the plots to prevent overlapping labels.

The axis ticks and labels can be supressed by setting the margins to a negative value. This is most useful when adding right-axes with `addXY` for example.

The value for `yright` can be set using the `setRtMargin` function if adding a plot using the secondary right axes; extract the fourth element of the returned value. The default is to set a narrow right-hand margin.

The value for `xtop` can be set to -2.2 if adding a plot using the secondary top axis. The default is to set the margin to 1.5, which allows only for a graph title.

Value

an object of class "Layout" with three named components and `num.graphs` numbered components:

`explanation` and each `numbered component`
: a list with two components:

`margin`
: the margin for the plot area

`fig`
: the figure area

`size`
: the size of the overall figure

`mat`
: the figure layout

Note

It is very easy to confuse the graph number, used by `setGraph` and the grid cell number referenced in `setlayout`. The grid cell number always ranges from 1 to the number of columns times the number of rows. The graph number ranges from 1 to `num.graphs` and skips the grid cell number if defined in the `explanation` argument. Printing the output object can help understand the graph layout.

There is nothing special about the cell allocated for the explanation; it has no special characteristics, therefore an explantion can be placed in any graph numbered cell and anything can be placed in the "explanation" cell. As an example, the "explanation" at the bottom of the figure can be used for a description of the figure that is more than one line in height.

See Also

`setPage`, `setGraph`, `setRtMargin`, `addTitle`

Examples

```
## Not run:
# See for examples of setLayout:
vignette(topic="BoxPlots", package="smwrGraphs")
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="PiperPlot", package="smwrGraphs")
demo(topic="HydroPrecip", package="smwrGraphs")
demo(topic="PiperScript", package="smwrGraphs")

## End(Not run)
```

---

setMargin *Graph Margins*

Description

Sets the margins for the plot area (support function).

Usage

```
setMargin(margin, yax, aux.label = FALSE, caption = TRUE)
```

Arguments

`margin`	incomplete plot margin specification, generally computed by `setGraph`.
`yax`	the y-axis information from a "pretty" function, required if the second entry of `margin` is `NA`.
`aux.label`	logical, if `TRUE`, then allocate space for a second level of x-axis labels.
`caption`	logical, if `TRUE`, then allocate space for figure caption?

Value

Complete plot margin specification.

See Also

setPage, setGraph, setLayout

---

setMultiPlot *Plot Parameters*

Description

Sets plot control list for individuals in data (support function).

Usage

```
setMultiPlot(current, Nobs = 1, name = "", what = "points",
  type = "solid", width = "standard", symbol = "circle",
  filled = TRUE, size = 0.09, color = "black", order = "as is")
```

Arguments

current	the plot parameters specified in the call to the high-level graphing function.
Nobs	the number of observations.
name	the name associated with the observation. See **Details**.
what	what kind of plot. Must be one of "points," symbols only; "lines," line segments connecting points only; "both," line segments connecting isolated symbols; "overlaid," line segments connecting points with symbols; "stairstep," stairstep line segments; or "vertical," vertical lines from the y-axis origin to the y value at each x value.
type	the type of line, if drawn. Must be one of "solid," "dashed," "dotted."
width	the width of the line, if drawn. Must be one of "standard," resulting in a line width of about 0.7 points; "color," resulting in a line width of about 0.8 points; "bold," resulting in a line width of about 1 point; or "hairline" resulting in a line width of about 0.5 points. Note these values are doubled if the font argument to setPage is "PPT."
symbol	type symbol, if drawn. Must be one of "circle;" "uptri," upward pointing triangle; "plus;" "x;" "diamond;" "downtri," downward pointing triangle; "square;" or "dot."
filled	logical, if TRUE, then fill the symbol. Valid only for symbol equal to "circle," "uptri," "diamond," "downtri," or "square."
size	the size of the symol in inches, if drawn.
color	the color of the plotted values. Can be a named color, such as "black" or "gray50" or an RGB color like "#4056FF."
order	specify the order of the symbols in the explanation. Can be "as is'–do nothing to order in explanation, "sort" or "increasing"—put into sorted order, "decreasing"—put in reverse order; or a vector theat specifies the exact order.

Details

Each of the arguments from `name` through `color` must have one entry for each observation. If a single value is given, then it is replicated for each observation. In general, it is convenient to set up a data frame with columns for group names with common values for the line or symbol. It is required that each group have common values for the line or symbol.

Value

A list having two components:

`current`	a list like `current` with the defaults set
`Explan`	a list for creating an explanation

See Also

`colors` for a list of color names, `setExplan`, for details about the list required for an explanation.

`setPlot` *Plot Parameters*

Description

Sets the plot control list (support function).

Usage

```
setPlot(current, name = "", what = "lines", type = "solid",
  width = "standard", symbol = "circle", filled = TRUE,
  size = 0.09, color = "black", area.color = NA, area.border = NA)
```

Arguments

`current`	list containing the current plot information or those requested by the user.
`name`	the name of the object plotted; used in the explanation. Expressions can be used, but if used, then `name` must be and expression for all calls that update the current plot information.
`what`	what to plot, see **Details**.
`type`	the line type, if drawn, must be one of "solid," "dashed," or "dotted."
`width`	the width of the line, if drawn, must be one of "standard;" "color," a little wider than "standard;" "bold," substantially wider than "standard;" or "hairline," used for ticks and borders.
`symbol`	the symbol to plot, if drawn, see **Details**.
`filled`	logical, if `TRUE` and a symbol is drawn, then fill with solid color.
`size`	the size of the symbol, in inches, if drawn.

color	the color of the symbol or line.
area.color	the color of a shaded area, required for completeness.
area.border	the boundary color of a shaded area, required for completeness.

Details

The value for what must be one of:

"points" symbols only,

"lines" lines only,

"both" lines connecting symbols with a small gap,

"overlaid" lines connecting symbols,

"stairstep" horizontal line to next x value with a vertical line to the y value,

"vertical" vertical lines from y equal 0 only.

"none" draw nothing

The value for symbol must be one of:

"circle" an open or filled circle, depending on filled,

"uptri" an open or filled up pointing triangle, depending on filled,

"plus" a plus sign (never filled),

"x" an x (never filled),

"diamond" an open or filled diamond shape, depending on filled,

"downtri" an open or filled down pointing triangle, depending on filled,

"square" an open or filled square, depending on filled,

"dot" a very small dot (never filled),

"+" a plus sign (never filled),

"none" no symbol or line.

Value

A list like current, but with the defaults supplied for any missing component.

Note

Vertical lines drawn by setting what to "vertical" are drawn from y equal 0 to each y value. The user will have more control over vertical lines by using the addBars function and setting the bar width to 0.

See Also

xyPlot, timePlot, qqPlot, piperPlot, probPlot, colorPlot, addBars

`setRtMargin` *Set Secondary Margin*

Description

Sets the right or top margin for graphs with secondary axes. Used After setting up the graphics environment, but before the call to the high-level graphics function to allocate space for an additional label and title.

Usage

```
setRtMargin(y, margin = c(NA, NA, NA, NA), right.labels = 7,
  right.log = FALSE, right.range = c(NA, NA))

setTopMargin(margin = c(NA, NA, NA, NA))
```

Arguments

`y`	the secondary y-axis data to be plotted, missing values are permitted and are ignored.
`margin`	incomplete plot margin specification, generally computed by `setGraph`.
`right.labels`	set up right-axis labels; the approximate number of labels.
`right.log`	logical, if `TRUE`, then log transform right axis.
`right.range`	set right-axis range.

Details

The values for `right.labels`, `right.log`, and `right.range` should be set exactly as in the call to `addXY`.

The `margin` is a numeric vector of length 4 specifying the bottom, left, top, and right margins around the plot, as decriibed in by the `mar` option in `par`. The function `setTopMargin` only changes the third value and the function `setRtMargin` only changes the fourth value. The value for `margin` is typically the output from `setGraph` or the defaults for these functions.

Value

The updated margin; only the right margin value is changed.

See Also

`setLayout`, `setGraph`, `addXY`

Examples

```
## Not run:
# See for examples of setRtMargin:
demo(topic="RightAxisExample", package="smwrGraphs")
# See for examples of setTopMargin:
demo(topic="TopAxisExample", package="smwrGraphs")

## End(Not run)
```

---

setSplom *Scatter Plot Matrix*

Description

Set up a scatter plot matrix.

Usage

```
setSplom(size = NULL, num.variables, show.all = FALSE,
  touching = TRUE, explanation = NULL, ymargin = 3.5)
```

Arguments

size	the width and height of the entire graph area, exclusive of explanation. If NULL, then use minimum of figure width and height.
num.variables	the number of variables to plot.
show.all	logical, if TRUE, then show the full grid. Otherwise only the lower triangular graphs.
touching	logical, if TRUE, then individual graphs touch. Otherwise a small gap separates individual graphs.
explanation	a description of where to place the explanation if needed. See **Details**.
ymargin	the left-margin for the plot area for the left column of graphs.

Details

If an explanation is needed, then explanation is used to indicate where the explanation is to be placed. The explanation can be placed either to the right of the grid of graphs, at the bottom of the grid, or in one of the grid cells.

To place an explanation to the right of the graphs, explantion should be set to list(right=ewid), where ewid is the width of the explanation. In this case, the total of width and ewid must be less than the total available for the page.

To place an explanation at the bottom of the graphs, explantion should be set to list(bottom=ehei), where ehei is the height of the explanation. In this case, the total pf height and ehei must be less than the total available for the page.

To place an explanation within a cell of the grid, `explantion` should be set to `list(grid=enum)`, where `enum` is the cell number in the grid. Cell numbers are sequential starting in the upper left and increasing by column. In this case `num.graphs` must be set to some number less than `num.cols` times `num.rows`.

Value

a list like `setLayout` with three additional components: `show.all`, `touching`, and `num.variables` from the call to `setSplom`.

See Also

`setLayout`

Examples

```
## Not run:
# A simple example
library(smwrData)
data(IonBalance)
setGD() #  set up a simple graphics page
AA.lo <- with(IonBalance, setSplom(num.variables=3, touching=FALSE))
with(IonBalance, splomPlot(cbind(Ca, Mg, Na), Panel=list(line="slr"), layout=AA.lo))
# See for another example of setSplom:
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

---

smwr.colors *Generate a Range of Colors*

Description

Generates a sequence of colors along a specified range.

Usage

```
greenRed.colors(n, alpha = 1)

redGreen.colors(n, alpha = 1)

blueRed.colors(n, alpha = 1)

redBlue.colors(n, alpha = 1)

warmCool.colors(n, alpha = 1)

coolWarm.colors(n, alpha = 1)

pastel.colors(n, alpha = 1)
```

Arguments

n	the number of colors to generate.
alpha	a measure of the intensity of the generated colors

Value

A sequence of character strings indicating the colors.

Note

`blueRed.colors` generates a sequence from blue to red through magenta. `redBlue.colors` generates a sequence from red to blue through magenta. `coolWarm.colors` generates a sequence from blue to red through green. `warmCool.colors` generates a sequence from red to blue through green. `greenRed.colors` generates a sequence from green to red through yellow. `redGreen.colors` generates a sequence from red to green through yellow. `pastel.colors` genereates a sequence of well-separated pastel colors useful for areas or bars.

See Also

`rainbow`, `hcl`

Examples

```
## Not run:
redGreen.colors(2)
blueRed.colors(2)
# For examples of warmCool.colors in graphs see
demo(topic="DurationHydrograph", package="smwrGraphs")
# All have similar usage

## End(Not run)
```

---

splomPlot *Scatter Plot Matrix*

Description

Produces a matrix of scatter plots

Usage

```
splomPlot(x, layout, Plot = list(name = "", what = "points", type =
  "solid", width = "standard", symbol = "circle", filled = TRUE, size =
  0.05, color = "black"), Panel = list(), axis.log = FALSE,
  axis.range = c(NA, NA), labels = 5, caption = "")
```

Arguments

x	the data to plot, must be either a matrix or a data frame.
layout	the output from setSplom
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
Panel	a list or a function, see **Details**.
axis.log	log-transform the x and y axis for all plots.
axis.range	set x- and y-axis ranges for all plots.
labels	set the number of labels for both the x and y axes. See linearPretty for details.
caption	the figure caption.

Details

Panel may be a tagged list, with any one of these options:
loess=span, where span is the span argument to loess.smooth;
line=opt, where opt='slr' for simple linear regression, or 'loc' for line of organic correlation, or '1:1' for the 1:1 line.
The format of the lines is taken from Plot.

Panel may also be a function with 3 arguments, x, y, and current, that adds to the plot and returns updated plot information. The function is called for each individual plot.

Value

Information about the graph.

Note

A call must be made to setPage and to setSplom to set up the graphics environment before calling splomPlot.

See Also

setPage, setSplom

Examples

```
## Not run:
# See for examples of splomPlot:
vignette(topic="LineScatter", package="smwrGraphs")

## End(Not run)
```

stiffPlot *Stiff Diagrams*

Description

Adds a Stiff diagram to an existing graph or produces a tabular presentation of Stiff diagrams in a graph.

Usage

```
stiffPlot(cations, anions, Stiff = list(fill = "gray50", outline =
  "black", height = 2/3, bar = "black"), yaxis.orient = "table",
  yaxis.order = "none", xaxis.range = c(NA, NA), ylabels = "Auto",
  xlabels = 7, catlabels = "Auto", anlabels = "Auto",
  xtitle = "Milliequivalents per liter", ytitle = "", caption = "",
  margin = c(NA, NA, NA, NA), ...)

addStiff(x, y, width, height, cations, anions, Stiff = list(fill =
  "gray50", outline = "black", height = 2/3, bar = "black"),
  xaxis.range = c(NA, NA), catlabels = "", anlabels = "",
  current = list(yaxis.log = FALSE, yaxis.rev = FALSE, xaxis.log =
  FALSE))
```

Arguments

cations	a matrix of cation data. Each row corresponds to the respective x and y value. Missing values are not permitted for addStiff, but are permitted for stiffPlot and result in no Stiff Diagram for that entry.
anions	a matrix of anion data. Each row corresponds to the respective x and y value. Missing values are not permitted for addStiff, but are permitted for stiffPlot and result in no Stiff Diagram for that entry.
Stiff	a list describing the Stiff diagram. See **Details**.
yaxis.orient	orientation of the y-axis values, must be either "table" or "grid." "Table" is sorted from top to bottom, "grid" is sorted from bottom to top.
yaxis.order	the order of the y-axis values, must be one of "none," "ascending," or "descending."
xaxis.range	the range of the x-axis corresponding to width in the call to addStiff or the range of the x-axis in the call to stiffPlot. See **Details**.
ylabels	set up y-axis labels.
xlabels	set up x-axis labels.
catlabels	labels for the values of the cations. For addStiff, the labels are applied to each Stiff diagram and for stiffPlot, the labels are stored and drawn on the explanation addExplanation.

anlabels	labels for the values of the anions. For addStiff, the labels are applied to each Stiff diagram and for stiffPlot, the labels are stored and drawn on the explanation addExplanation.
xtitle	x-axis title (also called x-axis caption).
ytitle	y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	not used, required for other methods.
x	the x-coordinates to place the center of each Stiff diagram. Missing values are permitted, but result in no Stiff diagram.
y	the y-coordinates to place the center of each Stiff diagram. Missing values are permitted, but result in no Stiff diagram.
width	the width in inches of the Stiff diagrams.
height	the height in inches of the Stiff diagrams.
current	the current plotting parameters. Typically, this would be the output from one of the graph creation functions like xyPlot.

Details

The units of the cation and anion data are generally in milli-equivalents per liter.

The Stiff argument must be a tagged list with these components:

fill the name of the color to fill each Stiff diagram. Must be a valid color name.

outline the name of the color to draw the outline or border for each Stiff diagram. Must be a valid color name.

height the height of each Stiff diagram, proportional to the overall height for each Stiff diagram.

bar the color of the central bar. May be "none" for no central bar.

The values for axis.range must be expressed as a negative value for the cation data, left-hand side of the diagram, and a positive value for anion data, right-hand side of the diagram.

Value

Information about the graph.

References

Hem J.D., 1989, Study and interpretation of the chemical characteristics of natural water: U.S. Geological Survey Water-Supply Paper 2254, 263 p.

See Also

addStiff, xyPlot

Examples

```
## Not run:
# See for examples of stiffPlot:
vignette(topic="PiperPlot", package="smwrGraphs")
# See for examples of addStiff:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

`strip.blanks` *Remove Spaces*

Description

Removes leading and trailing blanks from a character string.

Usage

```
strip.blanks(x)
```

Arguments

`x`	a character vector.

Value

A vector like `x`, but with leading and trailing spaces removed from each element.

See Also

`link[base]{sub}`

Examples

```
strip.blanks("   keep me   ")
```

surfacePlot *Surface plot*

Description

Creates a surface plot to show three-dimensional data or a colored surface plot to show four-dimensional data.

Usage

```
surfacePlot(pre, z.color = "lightblue", Surface = list(name = "",
  lineColor = "black", levels = 20, ramp = "coolWarm"), xtitle = "",
  ytitle = "", ztitle = "", margin = c(NA, NA, NA, NA),
  caption = "")
```

Arguments

pre	the output from preSurface.
z.color	the surface color. Can be either a character string indicating the color of the surface, or a numeric matrix. If a matrix, then must have length(x) rows and length(y) columns or length(x) - 1 rows and length(y)-1 columns. Missing values are permitted, but result in blank areas on the surface. See **Details**.
Surface	control parameters for the surface. See **Details**.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
ztitle	the z-axis title (also called z-axis caption).
margin	set up the plot area margins. To allocate space for a graph title, set the third value to 1.5, otherwise all values should be NA or 0.
caption	the figure caption.

Details

If z.color is a numeric matrix, then the values represent the average surface of the grid defined by x and y, that is it represents the value one-half way between each value of x and one-half way between each value of y, the mid point. If it has the same dimensions as z.surf, then the data are resampled by averaging each of the four corners to compute the mid point value. The values of z.color are assigned using the controls in Surface.
The Surface argument must be a tagged list with these components:

name the name of z.color used in the explanation.

lineColor the color for each line on the surface. If "none," then lines are not drawn.

levels the levels of the surface colors. May be either a single numeric value that indicates the approximate number of levels, or a vector that indicates the exact breaks in the levels.

ramp the name of the color ramp. May be "gray" or "grey" for a gray scale or the prefix name of a function that creates a range of colors, see coolWarm.colors for examples.

Value

Information about the graph.

Note

A call must be made to `setPage` or `setPDF` to set up the graphics environment before calling `surfacePlot`.

See Also

`setPage`, `preSurface`, `persp`

Examples

```
## Not run:
# See for examples of surfacePlot:
vignette(topic="GraphGallery", package="smwrGraphs")

## End(Not run)
```

---

ternaryPlot *Ternary Diagram*

Description

Produces a ternary diagram, also called a trilinear or triangular diagram.

Usage

```
ternaryPlot(x, y, z, Plot = list(name = "", what = "points", type =
  "solid", width = "standard", symbol = "circle", filled = TRUE, size =
  0.09, color = "black"), axis.range = c(0, 100), num.labels = 6,
  ticks = TRUE, grids = !ticks, orient = "c",
  xtitle = deparse(substitute(x)), ytitle = deparse(substitute(y)),
  ztitle = deparse(substitute(z)), units.title = "Percent",
  caption = "", margin = c(NA, NA, NA, NA))
```

Arguments

`x`	the x-axis (bottom) data.
`y`	the y-axis (left side) data.
`z`	the z-axis (right side) data. Note that `x`, `y`, and `z` do not need to sum to the axis range.
`Plot`	control parameters of the plot, see `link{setMultiPlot}` and **Details** for details.
`axis.range`	the range of the axes. Must be either c(0, 1) or c(0, 100).

num.labels	the number of labels to draw on each axis. Best selections are 2 giving (0, 100), 3 (0, 50, 100), 5 (0, 25, 50, 75, 100), or 6 (o, 20, 40, 60, 80, 100).
ticks	logical, if TRUE, then draw ticks.
grids	logical, if TRUE, then draw grid lines.
orient	the orientation of the graph. Must be "c" for clockwise or "a" for anti- or counter-clockwise
xtitle	title (also called caption) for the x-axis.
ytitle	title (also called caption) for the y-axis.
ztitle	title (also called caption) for the z-axis.
units.title	the units titles, should be either "Percent" of "Proportion" depending on axis.range.
caption	the figure caption.
margin	set up the plot area margins— ignored, included for consistency with other plotting functions in this package.

Details

The what component of the Plot argument must be either "points" or "none."

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling piperPlot.

References

Lorenz D.L., 2015, smwrGraphs

See Also

setPage, setMultiPlot, piperPlot, addTernary

Examples

```
## Not run:
# See for examples of ternaryPlot:
vignette(topic="PiperPlot", package="smwrGraphs")

## End(Not run)
```

`ternarySubplot` *Piper Diagram*

Description

Plots the trilinear diagram in the Piper diagram (support function).

Usage

```
ternarySubplot(x, y, z, what = "points", symbol = rep(1, length(x)),
  color = rep(1, length(x)), size = rep(0.05, length(x)),
  axis.range = c(0, 100), num.labels = 6, ticks = FALSE,
  grids = !ticks, orient = "c", xtitle = "x", ytitle = "y",
  ztitle = "z", plot = TRUE)
```

Arguments

`x`	the x-ccordinate values.
`y`	the y-coordinate values.
`z`	the z-coordinate values.
`what`	the type of plot, must be either "points," "lines," or "none."
`symbol`	the symbol to use if `what` is "points."
`color`	the color of the plot.
`size`	the size of the symbol if `what` is "points."
`axis.range`	the range of the axes. Must be either c(0, 1) or c(0, 100).
`num.labels`	the number of labels to draw on each axis.
`ticks`	logical, if `TRUE`, then draw ticks.
`grids`	logical, if `TRUE`, then draw grid lines.
`orient`	a single character, "c" indicates clockwise orientation for `x`, `y`, and `z` anything else indicates counter-clockwise.
`xtitle`	the x-axis title (also called x-axis caption).
`ytitle`	the y-axis title (also called y-axis caption).
`ztitle`	the z-axis title (also called z-axis caption).
`plot`	plot the data?

Details

Support function, to be called only from `piperPlot`.

Value

If `plot` is TRUE, then the range of the user coordinates. Otherwise, the `x`, `y`, and `z` values are converted to 2-dimensional values.

See Also

piperPlot

---

timePlot *Time-series Plots*

Description

Creates a plot of time-series data.

Usage

```
timePlot(x, y, Plot = list(), yaxis.log = FALSE, yaxis.rev = FALSE,
  yaxis.range = c(NA, NA), xaxis.range = range(x, na.rm = TRUE),
  ylabels = 7, xlabels = "Auto", xtitle = "", ytitle = "",
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'Date,numeric'
timePlot(x, y, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.range = range(x,
  na.rm = TRUE), ylabels = 7, xlabels = "Auto", xtitle = "",
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA), ...)

## S4 method for signature 'POSIXt,numeric'
timePlot(x, y, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.range = range(x,
  na.rm = TRUE), ylabels = 7, xlabels = "Auto", xtitle = "",
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA), ...)

## S4 method for signature 'numeric,numeric'
timePlot(x, y, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.range = range(x,
  na.rm = TRUE), ylabels = 7, xlabels = "Auto", xtitle = "",
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA), ...)

## S4 method for signature 'integer,numeric'
timePlot(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
```

```
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.range = range(x,
  na.rm = TRUE) + c(-1, 1), ylabels = 7, xlabels = "Auto",
  xtitle = "", ytitle = deparse(substitute(y)), caption = "",
  margin = c(NA, NA, NA, NA), xlabels.rotate = FALSE, ...)

## S4 method for signature 'difftime,numeric'
timePlot(x, y, Plot = list(name = "", what =
  "lines", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.range = range(x,
  na.rm = TRUE), ylabels = 7, xlabels = "Auto", xtitle = "Auto",
  ytitle = deparse(substitute(y)), caption = "", margin = c(NA, NA,
  NA, NA), ...)
```

Arguments

x	the time/date data.
y	the y-axis data.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.rev	logical, if TRUE, then reverse the y axis.
yaxis.range	set the range of the y-axis. See **Details**.
xaxis.range	set the range of the x-axis. Set at January 1 through December 31 for seasonPlot.
ylabels	set up y-axis labels. See linearPretty for details.
xlabels	set up x-axis labels. See **Details** for details for valid values.
xtitle	the x-axis title (also called x-axis caption). Generally should be blank as titles are typically set up by the axis labeling routines.
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	arguments for specific methods.
xlabels.rotate	logical, if TRUE, then rotate the x-axis labels so that they are perpendicular to the x-axis.

Details

For the timePlot methods where the time/date data are of class "Date," "POSIXt," or "numeric," the values for xlabels must be one of "hours," "days," "months," "years," "water years," or "Auto," which will select an appropriate axis labeling scheme based on the time span of the data. May also be a list of valid arguments to datePretty for finer control.

For the `timePlot` method where the time/date data are of class "integer," the value for `xlabels` must be one of "Auto," a number indicating the approximate number of labels, or a list of valid arguments to `linearPretty` for finer control.

For the `timePlot` method where the time/date data are of class "difftime," the value for `xlabels` must be one of "Auto" or a number indicating the approximate number of labels.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph.

Methods

signature(x = "Date", y = "numeric") Create a time-series plot for Date and numeric data.

signature(x ="POSIXt", y = "numeric") Create a time-series plot for POSIXt and #' numeric data.

signature(x = "numeric", y = "numeric") Create a time-series plot for dates in decimal format and numeric data.

signature(x = "integer", y = "numeric") Create a time-series plot for annual summaries of numeric data.

signature(x = "difftime", y = "numeric") Create a time-series plot for difftime and numeric data.

Note

The function `timePlot` produces a time-series plot. The function `seasonPlot` produces a plot of the annual cycle. There is no function in the smwrGraphs package that will automatically transform time/date data to the correct seasonal value; use `dectime(x) - trunc(dectime(x))`, where `x` is the time/date variable.

See Also

`setPage`, `xyPlot`, `seasonPlot`

Examples

```
## Not run:
# the months function is in lubridate
X <- as.Date("2001-01-15") + months(0:11)
set.seed(1)
Y <- runif(12)
setGD()
timePlot(X, Y)
# For more details of timePlot see
vignette(topic="DateAxisFormats", package="smwrGraphs")
```

```
vignette(topic="LineScatter", package="smwrGraphs")
demo(topic="AnnualFlowBarChart", package="smwrGraphs")
demo(topic="DurationHydrograph", package="smwrGraphs")
demo(topic="HydroPrecip", package="smwrGraphs")
demo(topic="RightAxisExample", package="smwrGraphs")

## End(Not run)
```

`timePretty` *Pretty Axis*

Description

Constructs information for making a nicely formatted date/time axis.

Usage

```
timePretty(x, labels = "Auto")
```

Arguments

`x`	time difference data
`labels`	either "Auto," which lets the function decide how many labels, the approximate number of labels, or the actual labels to use.

Value

Information about the axis labels.

See Also

`timePlot`

`transData` *Transform Data*

Description

Transforms numeric data to match any axis scaling (support function).

Usage

```
transData(data, logT = FALSE, revT = FALSE, trans = as.vector,
  transarg = NULL)
```

Arguments

data	data for axis
logT	logical, if TRUE, then log transform data
revT	logical, if TRUE, then reverse data to match the axis.
trans	arbitrary transform function to apply to data.
transarg	list of arguments to trans.

Value

A vector like data transformed to plot correctly on an axis.

See Also

transPlot, probPlot

---

transPlot *X-Y Plot*

Description

Creates an x-y plot using arbitrary monotonic transforms for the axes.

Usage

```
transPlot(x, xtrans, xinv, xtargs = NULL, y, ytrans, yinv,
  ytargs = NULL, Plot = list(name = "", what = "points", type =
  "solid", width = "standard", symbol = "circle", filled = TRUE, size =
  0.09, color = "black"), yaxis.range = c(NA, NA), xaxis.range = c(NA,
  NA), ylabels = "Auto", xlabels = "Auto",
  xtitle = deparse(substitute(x)), ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA))
```

Arguments

x	the x-axis data.
xtrans	the transformation function for the x-axis.
xinv	the inverse transformation for the x-axis.
xtargs	additional arguments to xtrans and xinv, as a list if necessary, NULL otherwise.
y	the y-axis data.
ytrans	the transformation function for the y-axis.
yinv	the inverse transformation for the y-axis.
ytargs	additional arguments to ytrans and yinv, as a list if necessary, NULL otherwise.

Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
yaxis.range	set the range of the y-axis.
xaxis.range	set the range of the x-axis.
ylabels	set up y-axis labels. See transPretty for details.
xlabels	set up x-axis labels. See transPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.

Value

Information about the graph.

Note

A call should be made to setPage to set up the graphics environment before calling transPlot.

See Also

setPage, transPretty

Examples

```
## Not run:
X <- seq(.25, 9.75, by=.25)
setGD()
# The Box-Cox transform (power of 1.5)
# The labels represent the original values; the line represents the transformed value
transPlot(X, I, I, y=X, ytrans=boxCox, yinv=IboxCox,
  ytargs=list(lambda=1.5, GM=1), Plot=list(what="lines"))
# For more details of transPlot see
demo(topic="MeasurementRating", package="smwrGraphs")

## End(Not run)
```

---

transPretty *Pretty Axis*

Description

Constructs information for making a nicely formatted numeric axis.

Usage

```
transPretty(x, hard = FALSE, labels = 11, style = "none",
  func = log, Ifunc = exp, ...)
```

Arguments

x	data defining the range to be plotted on the axis. Missing value are permitted, but ignored.
hard	logical, if TRUE, then use the minimum and maximum of x as the fixed range of the axis, otherwise find "nice" limits.
labels	either "Auto," which lets the function decide how many labels, the approximate number of labels, or the actual labels to use.
style	a character string indicating the style of the axis labels if they are not specifically listed in labels.
func	the forward transform function.
Ifunc	the backward (inverse) transform function.
...	additional arguments to func and Ifunc.

Value

Information about the axis lables.

See Also

transPlot

---

xyPlot *Plot Data*

Description

Creates a line/scatter plot.

Usage

```
xyPlot(x, y, Plot = list(), yaxis.log = FALSE, yaxis.rev = FALSE,
  yaxis.range = c(NA, NA), xaxis.log = FALSE, xaxis.range = c(NA,
  NA), ylabels = 7, xlabels = 7, xtitle = "", ytitle = "",
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'numeric,numeric'
xyPlot(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
```

```
  xaxis.range = c(NA, NA), ylabels = 7, xlabels = 7,
  xtitle = deparse(substitute(x)), ytitle = deparse(substitute(y)),
  caption = "", margin = c(NA, NA, NA, NA), ...)

## S4 method for signature 'factor,numeric'
xyPlot(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = c(NA, NA), ylabels = 7, xlabels = "Auto",
  xtitle = "", ytitle = deparse(substitute(y)), caption = "",
  margin = c(NA, NA, NA, NA), xlabels.rotate = FALSE, ...)

## S4 method for signature 'character,numeric'
xyPlot(x, y, Plot = list(name = "", what =
  "points", type = "solid", width = "standard", symbol = "circle", filled =
  TRUE, size = 0.09, color = "black"), yaxis.log = FALSE,
  yaxis.rev = FALSE, yaxis.range = c(NA, NA), xaxis.log = FALSE,
  xaxis.range = c(NA, NA), ylabels = 7, xlabels = "Auto",
  xtitle = "", ytitle = deparse(substitute(y)), caption = "",
  margin = c(NA, NA, NA, NA), xlabels.rotate = FALSE, ...)
```

Arguments

x	the x-axis data to plot.
y	the y-axis data to plot.
Plot	parameters defining the characteristics of the plot. See setPlot for a description of the parameters.
yaxis.log	logical, if TRUE, then log-transform the y axis.
yaxis.rev	logical, if TRUE, then reverse the y axis.
yaxis.range	set the range of the y-axis. See **Details**.
xaxis.log	logical, if TRUE, then log-transform the x axis.
xaxis.range	set the range of the x-axis. See **Details**.
ylabels	set up y-axis labels. See linearPretty for details.
xlabels	set up x-axis labels. See linearPretty for details.
xtitle	the x-axis title (also called x-axis caption).
ytitle	the y-axis title (also called y-axis caption).
caption	the figure caption.
margin	set the plot area margins, in units of lines of text. Generally all NA or the output from setGraph if appropriate.
...	additional arguments for specific methods.
xlabels.rotate	logical, if TRUE, then rotate x-axis labels 90 degrees (perpendicular to the axis).

Details

Setting `ylabels` or `xlabels` to 0 or negtive values will suppress ticks and labels. If negative, then try to create that absolute value number of labels. That can be useful for relative axes or specialized labeling.

For linear axes, the range can be set to virtually any pair of values. For log axes, the choice of range is more resticted—for less than one log-cycle, powers of whole numbers can be used; from 1 to about 3 log cycles, the choces should be powers of 3 or 10; and for more than 3 log cycles, the range sould be expressed only in powers of 10.

Value

Information about the graph

Methods

signature(x = "numeric", y = "numeric") Create a line or scatter plot from numeric x and y data.

signature(x = "factor", y ="numeric")") Create a vertical dot plot. Also useful for setting up a bar chart for discrete x-axis values.

signature(x = "character", y ="numeric")") Create a vertical dot plot. Also useful for setting up a bar chart for discrete x-axis values.

Note

A call should be made to `setPage` to set up the graphics environment before calling `xyPlot`.

See Also

`setPage`, `timePlot`, `colorPlot`

Examples

```
## Not run:
set.seed(1)
X <- rnorm(32)
Y <- X + rnorm(32)
setGD()
AA.pl <- xyPlot(X, Y, Plot=list(color="cyan4"))
# For more details of xyPlot see
vignette(topic="GraphAdditions", package="smwrGraphs")
vignette(topic="GraphGallery", package="smwrGraphs")
vignette(topic="GraphSetup", package="smwrGraphs")
vignette(topic="LineScatter", package="smwrGraphs")
demo(topic="Coplot-complexScatterPlot", package="smwrGraphs")
demo(topic="TopAxisExample", package="smwrGraphs")

## End(Not run)
```

Index