Package: kwb.plot (via r-universe)

August 22, 2024

Title some useful functions for plotting

Version 0.5.0

Description Some useful functions for plotting.

License MIT + file LICENSE

URL https://github.com/kwb-r/kwb.plot

BugReports https://github.com/kwb-r/kwb.plot/issues

Imports ggplot2, gridExtra, kwb.utils, rlang

Suggests devtools, knitr, kwb.datetime, rmarkdown, testthat

VignetteBuilder knitr

Remotes github::kwb-r/kwb.datetime, github::kwb-r/kwb.utils

Encoding UTF-8

RoxygenNote 7.2.3

Repository https://kwb-r.r-universe.dev

RemoteUrl https://github.com/KWB-R/kwb.plot

RemoteRef HEAD

RemoteSha 5548903e51b0337b0043d21a091855c181a3c9d0

Contents

.defaultPlotParameter *Default Plot Parameters for Hydraulic Variables*

Description

Default Plot Parameters for Hydraulic Variables

Usage

```
.defaultPlotParameter(parameterName, variableName)
```

Arguments

parameterName	one of c("type", "col", "ylab", "pch")
variableName	variable name matching "(Q|H|v)[.](raw|signal|interpol|pred)"

addLabels *Add Labels*

Description

add labels at given x-positions to plot (with alternating y positions to avoid overlapping labels)

Usage

```
addLabels(
  x,
  labels = as.character(x),
  y0 = 0,
  bandheight = 0.1,
  col = "black",
  group.size = 3,
  alternating = FALSE,
  col.line = "black",
  lty = 1,
  lty.horiz.line = 0,
  adj = -0.1,
  cex = 0.7
)
```

Arguments

x	x positions of the labels
labels	vector of character containing the labels
y0	base y position of the labels
bandheight	height of band "around" (alternating == FALSE) or above (alternating == TRUE) y0 as a fraction of the plot region height (e.g. 0.1 for 10 percent). Default: 0.1
col	colour of the labels
group.size	passed to labelPositionY
alternating	passed to labelPositionY
col.line	colour of the lines to the labels
lty	type of the lines to the labels (as defined in par)
lty.horiz.line	type of the horizontal line (as defined in par)
adj	passed to text
cex	passed to text

---

addTimeAxis *Add Time Axis*

Description

Add Time Axis

Usage

```
addTimeAxis(
  myDateTime,
  xlim = range(myDateTime),
  n = 20,
  time.format = NULL,
  add.grid = FALSE,
  padj = 0.5
)
```

Arguments

myDateTime	vector of POSIXct timestamps
xlim	lower and upper limits of range of timestamps to be shown
n	number of timestamps to be shown, passed to pretty
time.format	time format string such as " format.POSIXct
add.grid	if TRUE vertical lines are added at the positions of the time tickmarks
padj	passed to axis

apply_elements_text *Apply themes with given text properties to a plot*

Description

Apply themes with given text properties to a plot

Usage

```
apply_elements_text(x, elements)
```

Arguments

x	ggplot object
elements	objects as created by function element_text

appropriateLimits *default limits for plotting values*

Description

default limits for plotting values

Usage

```
appropriateLimits(x, limits = NULL, default = c(0, 1))
```

Arguments

x	vector of numeric or POSIXt (min, max must be able to be applied).
limits	vector of two elements. Default: NULL.
default	vector of two elements. Default: c(0, 1)

Value

returns *limits* if *limits* is a vector of two non-NA values. If the first element of *limits* is NA it is replaced with the minimum of *x* (or with *default[1]* if the minimum is NA). If the second element of *limits* is NA it is replaced with the maximum of *x* (or with *default[2]* if the maximum is NA).

args_mybarplot *Create argument list for mybarplot*

Description

Create argument list for mybarplot

Usage

```
args_mybarplot(
  cex.axis = graphics::par("cex.axis"),
  cex.names = graphics::par("cex.axis"),
  ...,
  args.text = NULL,
  args.yaxis = NULL
)
```

Arguments

cex.axis	character expansion factor of axis labels
cex.names	character expansion factor of bar group labels
...	additional arguments to barplot
args.text	list of arguments that are given to text being called to plot the bar group labels. Use args_text to create such a list.
args.yaxis	list of arguments that are given to axis being called to plot the y axis. Use args_yaxis to create such a list.

args_text *Create argument list for text*

Description

Create argument list for text

Usage

```
args_text(y.abs = NULL, y.rel = -0.01, srt = 90, adj = c(1, 0.5), ...)
```

Arguments

y.abs	absolute y positions
y.rel	relative y positions
srt	see text
adj	see text
...	further arguments to text

Value

list of arguments that may be used as `args.text` in `mybarplot`

`args_yaxis` *Create argument list for axis*

Description

Create argument list for axis

Usage

```
args_yaxis(
  side = 2,
  at = graphics::axTicks(side),
  labels = paste0(at, unit),
  unit = "",
  las = 1,
  ...
)
```

Arguments

`side`	1: bottom, 2: left, 3: top, 4: right
`at`	where to put the ticks/labels
`labels`	vector of character labels
`unit`	text appended to the labels with `paste0`
`las`	numeric in 0,1,2,3; the style of axis labels (see `par`)
`...`	additional parameters passed to `axis`

Value

list of arguments that may be used as `args.yasis` in `mybarplot`

arrange_in_pdf *Arrange Plots in PDF file (DIN A4)*

Description

Arrange ggplot objects with gridExtra::grid.arrange and write the result to a PDF file in DIN A4 format

Usage

```
arrange_in_pdf(plots, landscape = TRUE, ...)
```

Arguments

plots	list of ggplot objects
landscape	passed to preparePdf
...	arguments passed to grid.arrange

arrange_in_png *Arrange Plots in PNG file*

Description

Arrange ggplot objects with gridExtra::grid.arrange and write the result to a PNG file

Usage

```
arrange_in_png(plots, filename, ...)
```

Arguments

plots	list of ggplot objects
filename	full path to the PNG file to be written. The extension ".png" will be automatically appended if it is omitted
...	arguments passed to grid.arrange

bestRowColumnSetting *best nrow/ncol setting for n plots*

Description

Number of rows and columns in an optimal plot grid for n plots

Usage

```
bestRowColumnSetting(n, target.ratio = 1, device.ratio = NULL)
```

Arguments

n	number of plots to be placed in a matrix of equally sized plot cells
target.ratio	desired height/width ratio within each plot (ignoring margins). Default: 1
device.ratio	desired height/width ratio in the output device. Default: kwb.plot:::getPlotRegionRatio()

Value

named vector of two elements with the first element (nrow) representing the number of rows and the second element (ncol) representing the number of columns for an optimal plot grid to be used for n plots in the current plot region

Examples

```
# save current graphical parameter setting
old.par <- graphics::par(no.readonly = TRUE)

for (i in 2:5) {

  graphics::par(mfrow = kwb.plot::bestRowColumnSetting(i))

  for (j in 1:i) {
    plot(seq_len(j), main = paste0("j = ", j, "/", i))
  }
}

# restore graphical parameter setting
graphics::par(old.par)
```

cmToUserWidthAndHeight *Convert Length in Centimetres to Lengths in User Coordinates*

Description

Convert Length in Centimetres to Lengths in User Coordinates

Usage

```
cmToUserWidthAndHeight(cm)
```

Arguments

cm	length in centimetres

demo.adj *Demo of Graphical Parameter "adj"*

Description

Demonstration of the effect of the graphical parameter "adj" (horizontal and vertical adjustment)

Usage

```
demo.adj(text = "Text", srt = c(0, 90), cex = 1, ..., to.pdf = FALSE)
```

Arguments

text	text to be plotted
srt	The string rotation in degrees (see par)
cex	character expansion factor
...	further arguments passed to text
to.pdf	if TRUE the output goes into a PDF file

Examples

```
kwb.plot::demo.adj()
kwb.plot::demo.adj("Exampletext", srt = 30)
#kwb.plot::demo.adj(srt = c(0, 45, 90), to.pdf = TRUE)
```

demo_themes_line *demo themes line*

Description

demo themes line

Usage

```
demo_themes_line()
```

demo_themes_rect *demo themes rect*

Description

demo themes rect

Usage

```
demo_themes_rect()
```

demo_themes_text *demo themes text*

Description

demo themes text

Usage

```
demo_themes_text()
```

demo_theme_properties *Plots Demonstrating Theme Properties*

Description

Plots Demonstrating Theme Properties

Usage

```
demo_theme_properties(x = example_plot_2(), to_pdf = TRUE)
```

Arguments

x	ggplot object on which to demonstrate the theme properties
to_pdf	if TRUE (default) the plots are written to a pdf file

drawBoxplot *Draw Boxplot Icon*

Description

draws a symmetric boxplot icon around a centre

Usage

```
drawBoxplot(
  centre.x,
  centre.y,
  boxwidth.cm = 1,
  boxheight.cm = boxwidth.cm,
  whisker.cm = boxheight.cm
)
```

Arguments

centre.x	x coordinate in user coordinates around which the box is to be drawn
centre.y	y coordinate in user coordinates around which the box is to be drawn
boxwidth.cm	width of the box in cm. Default: 1
boxheight.cm	height of the box in cm. Default: *boxwidth.cm*
whisker.cm	length of the whiskers in cm. Default: *boxheight.cm*

Examples

```
### prepare a simple plot area
plot(1:5)

### draw a box around the centre at (2, 2) with default proportions
drawBoxplot(2, 2, boxwidth.cm = 1)

### draw a box around the centre at (3, 3) with differing width and height
drawBoxplot(3, 3, boxwidth.cm = 2, boxheight.cm = 1)

### draw a box around the centre at (4, 4) with modified whisker lengths
drawBoxplot(4, 4, boxwidth.cm = 0.5, boxheight.cm = 1.5, whisker.cm = 0.5)
```

---

element_types *element types*

Description

element types

Usage

```
element_types(secondary = FALSE)
```

Arguments

secondary	if TRUE all parameters are returned otherwise those parameters related to the x axis on top, the y axis on the right or the distict x an y parameters related to the "stip.text" parameter are omitted

---

example_plot *Simple ggplot Barplot*

Description

Simple ggplot example barplot

Usage

```
example_plot()
```

example_plot_2 *Simple ggplot Scatter Plot*

Description

Simple ggplot Scatter Plot

Usage

```
example_plot_2(n = 7)
```

Arguments

n	number of points

generic_barplot *Create a generic barplot using ggplot.*

Description

This function generates a barplot using ggplot, providing flexibility in customising the plot according to your data. The barplot can represent different groups and sub-groups, with the option to calculate bar heights based on counts or specific values.

Usage

```
generic_barplot(
  data,
  group_by,
  values_in = NULL,
  fill_by = NULL,
  percentaged = FALSE,
  reverse = FALSE,
  args_geom_bar = list()
)
```

Arguments

data	A data frame containing the data for plotting.
group_by	The name of the column in data that contains the group labels. Each unique value in this column corresponds to a distinct group, and a bar will be generated for each group. The column must be of type character or factor.
values_in	(Optional) The name of the column in data that contains the values used to calculate the bar heights. If not provided, the bar heights will be calculated based on the counts, i.e., the number of rows in data that belong to each group as defined by the values in column group_by.

fill_by	(Optional) The name of the column in data that defines sub-groups. Each sub-group will be represented by a different fill colour, and the bars will consist of stacked, filled rectangles. This parameter is useful when visualising sub-groups within each group.
percentaged	If set to TRUE, all bars will be stretched to extend to the full height of the plot. This option is only applicable when fill_by is specified where each bar is represented by a stack of rectangles each of which is filled in a different colour. The heights of the rectangles will then reflect the proportions of each sub-group (count or sum of corresponding rows or values in the values_in column) within a group. The default value is FALSE.
reverse	Whether or not to reverse the stack order, passed to position_fill. The default is FALSE.
args_geom_bar	(Optional) List of arguments that are to be passed to geom_bar, such as width (bar width), fill (fill colour).

Details

Note: This description was written by the help of ChatGPT.

Value

A barplot visualising the data using ggplot.

Examples

```
# Basic usage
my_data <- data.frame(
 group = c("A", "A", "A", "B", "B", "C", "C"),
 value = c(10, 15, 3, 8, 12, 5, 20),
 subgroup = c("X", "Y", "X", "X", "Y", "X", "Y")
)

generic_barplot(
  data = my_data,
  group_by = "group",
  values_in = "value",
  fill_by = "subgroup"
)

# Percentage-based bar heights
generic_barplot(
  data = my_data,
  group_by = "group",
  values_in = "value",
  fill_by = "subgroup",
  percentaged = TRUE
)

# Pass arguments to geom_bar()
generic_barplot(
```

```
  data = my_data,
  group_by = "group",
  args_geom_bar = list(fill = "red", width = 0.5)
)
```

`getCurrentLimits` *Get Current xlim and ylim*

Description

get the values of xlim and/or ylim that created the last plot

Usage

```
getCurrentLimits(
  type = "xy",
  xaxs = graphics::par("xaxs"),
  yaxs = graphics::par("yaxs")
)
```

Arguments

`type`	one of "x", "y" or "xy", see section "Value"
`xaxs`	value of argument *xaxs* when calling a plot function last, see `par`
`yaxs`	value of argument *yaxs* when calling a plot function last, see `par`

Value

type = "x" or "y": vector of two elements corresponding to xlim or ylim, respectively. *type* = "xy": list with elements *xlim* and *ylim* each of which is a vector of two elements.

`getPlotRegionRatio` *Current plot region's ratio*

Description

Current plot region's height/width ratio

Usage

```
getPlotRegionRatio()
```

Value

numeric of length one representing the width/height ratio of the current plot region

`getPlotRegionSizeInCm` *Size of Current Plot Region in Centimetres*

Description

Size of Current Plot Region in Centimetres

Usage

```
getPlotRegionSizeInCm()
```

`getPlotRegionSizeInInches` *Size of Current Plot Region in Inches*

Description

Size of Current Plot Region in Inches

Usage

```
getPlotRegionSizeInInches()
```

`getPlotRegionSizeInPixels` *Size of Current Plot Region in Pixels*

Description

Size of Current Plot Region in Pixels

Usage

```
getPlotRegionSizeInPixels()
```

`getPlotRegionSizeInUserCoords` *Size of Current Plot Region in User Coordinates*

Description

Size of Current Plot Region in User Coordinates

Usage

```
getPlotRegionSizeInUserCoords()
```

Value

list with elements *width*, *height*, *left*, *bottom*, *right*, *top*

`ggplot_text_lines` *Plot Text Only*

Description

Plot Text Only

Usage

```
ggplot_text_lines(
  x,
  max_rows = max(10, length(x)),
  size = 3,
  max_chars = 170,
  margins_cm = c(0, 0, 0, 0)
)
```

Arguments

`x`	vector of character representing the rows of text to be plotted from top to bottom
`max_rows`	number of rows to be prepared in the plot area. Increase/decrease this number to decrease/increase the line spacing
`size`	text size. Default: 3
`max_chars`	maximum number of characters to be used in each row. Longer lines that shortened to this number using `shorten`
`margins_cm`	vector of four numerics giving the bottom, left, top and top margins in cm, respectively. Default: `c(0, 0, 0, 0)`

Examples

```
ggplot_text_lines(paste(c("first", "second", "third"), "row"))
```

`ggplot_themes` *List of available ggplot2-Themes*

Description

Return a list of predefined themes from the ggplot2-package

Usage

```
ggplot_themes()
```

`inchesToUserWidthAndHeight`
Convert Length in Inches to Lengths in User Coordinates

Description

Convert Length in Inches to Lengths in User Coordinates

Usage

```
inchesToUserWidthAndHeight(inches)
```

Arguments

`inches`	length in inches

inLimits *are values within limits (e.g. xlim, ylim)?*

Description

are values within limits (e.g. xlim, ylim)?

Usage

```
inLimits(x, limits)
```

Arguments

x	vector of values
limits	two-element vector of (lower and upper) limits

Value

vetor of logical. Each element corresponds to an element in *x* and is TRUE if the element is within the limits or FALSE if the element is out of the limits.

mybarplot *barplot with more flexibility on labels and axis*

Description

barplot function allowing to set label and axis arguments, e.g. rotation of labels, giving a unit to the axis values

Usage

```
mybarplot(
  height,
  names.arg = NULL,
  cex.axis = graphics::par("cex.axis"),
  cex.names = graphics::par("cex.axis"),
  ...,
  args.text = NULL,
  args.yaxis = NULL
)
```

Arguments

height	as in barplot
names.arg	as in barplot
cex.axis	character expansion factor of axis labels
cex.names	character expansion factor of bar group labels
...	additional arguments to barplot
args.text	list of arguments that are given to text being called to plot the bar group labels. Use args_text to create such a list.
args.yaxis	list of arguments that are given to axis being called to plot the y axis. Use args_yaxis to create such a list.

Value

Returns (invisibly) what barplot returns: the x positions of the bars that have been plotted

Examples

```
height1 <- structure(1:10, names = paste("Category", 1:10))

height2 <- matrix(
  1:12, nrow = 3, dimnames = list(
    c("A", "B", "C"), paste("Long cateogry name", 1:4))
)

graphics::par(mfrow = c(1, 2))

# In the simplest form, mybarplot does what barplot does...
x1 <- graphics::barplot(height1, main = "barplot")
x2 <- mybarplot(height1, main = "mybarplot")

# ... and gives the same result
identical(x1, x2)

# We cannot distinguish the labels. With mybarplot we have finer access to the
# labels, such as rotation, yposition and adjustment
graphics::par(mfrow = c(1, 3))

graphics::barplot(height1, las = 2, main = "barplot")

mybarplot(height1, las = 2, main = "mybarplot",
          args.text = args_text(srt = 45, y.abs = -1, col = "blue"))

mybarplot(height1, las = 2, main = "mybarplot",
          args.text = args_text(srt = 90, y.abs = 10, col = "red"))

# Concerning the y-axis, you may e.g. give a unit to the values or modify the
# positions at which to draw ticks
graphics::par(mfrow = c(1, 1))
```

```
mybarplot(height1, las = 2, cex.names = 0.8,,
          args.text = args_text(srt = 90, y.abs = -0.5),
          args.yaxis = args_yaxis(at = 0:10, unit = " %", col.axis = "blue"))

graphics::par(mfrow = c(1, 2))

graphics::barplot(height2, main = "barplot", cex.names = 0.8)

mybarplot(height2, main = "mybarplot", cex.names = 0.8,
          args.text = args_text(srt = 30))

graphics::barplot(height2, las = 1, main = "barplot",   beside = TRUE,
                  cex.names = 0.8)

mybarplot(height2, las = 1, main = "mybarplot", beside = TRUE,
          cex.names = 0.8, args.text = args_text(srt = 30))
```

`niceLabels` *Nice Labels*

Description

Generates a nice vector of labels by suppressing labels at certain positions. If a *labelStep* is given, only every *labelStep*-th label is considered. If a vector *labpos* of label positions and a minimum distance *mindist* of labels is given, it is guaranteed that the distance between labels to be shown is at least *mindist*.

Usage

```
niceLabels(label, labelstep = NULL, labelpos = NULL, mindist = 1, offset = 0)
```

Arguments

`label`	vector of labels
`labelstep`	step width in which labels are to be printed, e.g. labelStep = 2 is used for plotting every second label.
`labelpos`	label positions
`mindist`	minimum distance between labels
`offset`	offset by which the index of the first remaining label (default: 1) is shifted. Default: 0, i.e. the labels at indices *1*, *1 + 1*labelstep*, *1 + 2*labelstep*, etc. remain. Offset = 1: the labels at indices *2*, *2 + 1*labelstep*, *2 + 2*labelstep*, etc. remain.

Examples

```
x <- matrix(1:12, nrow = 3)

names.arg <- rep(1:4, each = 3)

mybarplot <- function(x, ...) {
  barplot(x, horiz = TRUE, las = 1, beside = TRUE, ...)
}

mybarplot(x, names.arg = names.arg)
mybarplot(x, names.arg = niceLabels(names.arg, labelstep = 3))
mybarplot(x, names.arg = niceLabels(names.arg, labelstep = 3, offset = 1))
mybarplot(x, names.arg = niceLabels(names.arg, labelstep = 3, offset = 2))
```

---

output_to_png *Redirect Output of Plot Function to PNG file*

Description

Redirect the output of a plot function to a PNG file

Usage

```
output_to_png(
  FUN,
  args,
  filename,
  size = unlist(kwb.utils::DIN.A4()),
  units = "cm",
  res = 300,
  ...,
  dbg = TRUE
)
```

Arguments

FUN	plot function to be called
args	list of arguments given to the plot function FUN
filename	full path to the PNG file to be written. The extension ".png" will be automatically appended if it is omitted
size	vector containing the width (first element) and height (second element) of the plot, as passed to png. By default the dimensions of a DIN A4 page are used.
units	units as passed to png
res	resolution as passed to png
...	further arguments passed to png
dbg	if TRUE a message "Plotting to <filename>..." is shown

plotLevelFrequencies *Plot Frequencies of Level Combinations*

Description

For combinations of columns c1 and c2 in data, ggplot-objects are generated each of which shows the number (relative = FALSE) or percentage (relative = TRUE) of the different possible combinations of values in c1, and c2. The combinations of column names are created from the permutation of column names given in vars_1 and vars_2, respectively.

Usage

```
plotLevelFrequencies(data, vars_1 = NULL, vars_2 = NULL, relative = TRUE)
```

Arguments

data	data frame
vars_1	first vector of column names
vars_2	second vector of column names
relative	if TRUE (default)

Value

list of objects of class "ggplot"

Examples

```
data <- data.frame(
  fruit = c("apple", "cherry", "apple", "banana", "cherry"),
  colour = c("green", "red", "red", "yellow", "green"),
  size = c("small", "small", "big", "small", "big")
)

plotLevelFrequencies(data)
```

plotRain *Plot Rain Data as Bars*

Description

Plot Rain Data as Bars

Usage

```
plotRain(
  timestamps,
  values,
  gaugeName = "",
  xlim = NULL,
  ylim = NULL,
  signal.width = NA,
  shift.to.begin = 0,
  col = "blue",
  pch = 16,
  reverse.ylim = TRUE,
  las = 1,
  innerMargins = c(0, 0, 0, 0),
  add.time.axis = TRUE,
  ...
)
```

Arguments

timestamps	timestamps
values	precipitation values in mm
gaugeName	Name of rain gauge to appear in the y label
xlim	x limits
ylim	y limits
signal.width	width of time intervals represented by timestamps, in seconds. If NA (default) this value is deduced from the time differences in *timestamps*
shift.to.begin	value in seconds by which the timstamps need to be shifted in order to get the begin of the time intervals that the timestamps represent. Default: 0 (meaning that the timestamps represent the beginnings of the time intervals). A value of *signal.width* would mean that the timestamps represent the ends of the time intervals and a value of *signal.width/2* would mean that the timestamps represent the midpoints of the time intervals that they represent.
col	fill colour of bars. Default: "blue"
pch	plot character to be used for plotting data points (at the given timestamps), additionally to plotting bars. Set to NA in order to suppress plotting these points
reverse.ylim	if TRUE (default), the y axis is reversed, i.e. the bars are heading from top to bottom
las	numeric in 0,1,2,3; the style of axis labels. See help for par.
innerMargins	passed to plot_variable
add.time.axis	passed to plot_variable
...	further arguments passed to points

plot_all_to_pdf *Print all ggplot Objects to a PDF file*

Description

Print all ggplot Objects to a PDF file

Usage

```
plot_all_to_pdf(plots, landscape = TRUE, ...)
```

Arguments

plots	list of ggplot objects
landscape	passed to preparePdf
...	further arguments passed to preparePdf

plot_curve_area *Plot Filled Area Below a Curve Line*

Description

Plot Filled Area Below a Curve Line

Usage

```
plot_curve_area(x, y, y.base = 0, col = NA, ...)
```

Arguments

x	vector of x coordinates
y	vector of y coordinates
y.base	y coordinate of horizontal line that closes the area
col	colour of area to be plotted. Default: NA
...	further arguments given to polygon such as border (colour of the border of the polygon)

Examples

```
x <- seq(-pi, pi, pi/8)
y <- sin(x)

plot(x, y)
plot_curve_area(x, y, 0, col = "red")

plot(x, y)
plot_curve_area(x, y, -1, col = "red")
plot_curve_area(x, y, 1, col = "green")
```

`plot_curve_areas_gg` *Plot Filled Areas below Curve Lines using ggplot*

Description

Plot Filled Areas below Curve Lines using ggplot

Usage

```
plot_curve_areas_gg(
  x = seq_along(y_list[[1]]),
  y_list,
  col = NULL,
  stack = FALSE,
  legend = TRUE,
  line_colour = "black"
)
```

Arguments

`x`	x positions of each curve given in `y_list`
`y_list`	list of vectors of y positions each of which must be as long as `x`
`col`	vector of the same length as `y_list` giving the colours of the areas to be filled
`stack`	if `TRUE` (the default if `FALSE`) the areas are stacked on top of each other otherwise they are overlayed.
`legend`	if `TRUE` (default) the standard legend is shown, else it is hidden
`line_colour`	colour of the curve lines

Examples

```
x <- 1:10
y_list <- list(rep(10, 10), 10*sin(x/pi), 5*cos(x/pi))

# Basic plot
```

```
plot_curve_areas_gg(x, y_list)

# Set the colours (must be as many as vectors in y_list)
plot_curve_areas_gg(x, y_list, col = c("black", "white", "red"))

# Hide the legend
plot_curve_areas_gg(x, y_list, legend = FALSE)

# Stack the values instead of overlaying them
plot_curve_areas_gg(x, y_list, stack = TRUE)
```

`plot_variable` *Plot a Variable of a Data Frame*

Description

Plot a Variable of a Data Frame

Usage

```
plot_variable(
  hydraulicData,
  variableName = names(hydraulicData)[[2]],
  type = .defaultPlotParameter("type", variableName),
  col = .defaultPlotParameter("col", variableName),
  ylab = .defaultPlotParameter("ylab", variableName),
  pch = .defaultPlotParameter("pch", variableName),
  xlim = NULL,
  ylim = NULL,
  add = FALSE,
  plot.grid = TRUE,
  innerMargins = c(0, 0, 0, 0),
  absolute = FALSE,
  reverse.ylim = FALSE,
  add.time.axis = TRUE,
  time.axis.in.margins = FALSE,
  ...
)
```

Arguments

`hydraulicData`	data frame with at least two columns. First column is expected to contain the timestamps
`variableName`	default: name of second column
`type`	plot type, passed to `points`
`col`	colour, passed to `points`

ylab	y label, passed to plot
pch	plot character, passed to points
xlim	x limits, passed to plot
ylim	y limits, passed to plot
add	if TRUE points are added to an existing plot
plot.grid	if TRUE a grid is drawn
innerMargins	margins (bottom, left, top, right) within the plot area
absolute	if TRUE, innerMargins are to be interpreted as absolute values instead of fractions of the plot region. Default: FALSE
reverse.ylim	if TRUE (default), the y axis is reversed, i.e. the bars are heading from top to bottom
add.time.axis	if TRUE a time axis is drawn
time.axis.in.margins	if TRUE the time axis is drawn only within the plot area excluding the margins
...	additional arguments passed to plot

`preview_themes` *Preview the Effects of Themes on a Plot*

Description

Preview the effects of ggplot2-themes on a given plot

Usage

```
preview_themes(
  x = example_plot(),
  themes = ggplot_themes(),
  to_pdf = TRUE,
  landscape = TRUE,
  ...
)
```

Arguments

x	ggplot object
themes	list of ggplot-themes as returned by theme
to_pdf	if TRUE (default) the output goes to a PDF file
landscape	if TRUE (default) the output to a PDF file will be in DIN A4 landscape format, else in DIN A4 portrait format
...	arguments passed to grid.arrange

setMargins *Set the Plot Margins*

Description

Set the Plot Margins

Usage

```
setMargins(bottom = NA, left = NA, top = NA, right = NA)
```

Arguments

bottom	bottom margin as used in par("mar")
left	left margin as used in par("mar")
top	top margin as used in par("mar")
right	right margin as used in par("mar")

set_labels *Set the Labels in a List of ggplot Objects*

Description

Set the Labels in a List of ggplot Objects

Usage

```
set_labels(
  plots,
  ...,
  indices = seq_along(plots),
  label_data = NULL,
  action = c("replace", "append", "prepend")[1],
  sep = " "
)
```

Arguments

plots	list of ggplot objects as returned by ggplot
...	name-value pairs as given to labs. Possible names are e.g. "x", "title", "subtitle", "caption". The values are vectors of character that are recycled to the length of indices. They are used as labels given to each plot or to the plots selected by their indices.

indices	indices of the plots to which the label is to be given. By default the label is given to all plots
label_data	data frame containing the different label types in different columns. If given and not NULL arguments in ... are ignored.
action	one of "replace" (replace the existing label), "append" (append to the existing label), "prepend" (prepend to the existing label).
sep	separator to be used when append is one of "append", "prepend".

Examples

```
p <- example_plot_2()

plots <- list(p, p, p, p)

plots_1 <- set_labels(
  plots, title = c("Title A", "Title B", "Title C", "Title D"),
  subtitle = "same subtitle", x = c("x label one", "x label two")
)

plots_2 <- set_labels(
  plots, title = c("(A)", "(B)", "(C)", "(D)"),
  subtitle = "(always the same)", x = c("(one)", "(two)"),
  action = "append"
)

label_data <- data.frame(title = "Titles set with label_data",
  subtitle = sprintf("Plot %d", seq_along(plots))
)
plots_3 <- set_labels(plots, label_data = label_data)

do.call(gridExtra::grid.arrange, plots_1)
do.call(gridExtra::grid.arrange, plots_2)
do.call(gridExtra::grid.arrange, plots_3)
```

---

set_subtitles *Set the Subtitles in a List of ggplot Objects*

Description

Set the Subtitles in a List of ggplot Objects

Usage

```
set_subtitles(plots, subtitle, indices = seq_along(plots), ...)
```

Arguments

plots	list of ggplot objects as returned by ggplot
subtitle	subtitle (character) to be given to each plot or to the plots selected by their indices
indices	indices of the plots to which the subtitle is to be given. By default the subtitle is given to all plots
...	additional arguments to set_labels, such as action

---

set_titles *Set the Titles in a List of ggplot Objects*

Description

Set the Titles in a List of ggplot Objects

Usage

```
set_titles(plots, title, indices = seq_along(plots), ...)
```

Arguments

plots	list of ggplot objects as returned by ggplot
title	title (character) to be given to each plot or to the plots selected by their indices
indices	indices of the plots to which the title is to be given. By default the title is given to all plots
...	additional arguments to set_labels, such as action

---

set_xlabs *Set the x Axis Label in a List of ggplot Objects*

Description

Set the x Axis Label in a List of ggplot Objects

Usage

```
set_xlabs(plots, xlab, indices = seq_along(plots), ...)
```

Arguments

plots	list of ggplot objects as returned by ggplot
xlab	x axis label (character) to be given to each plot or to the plots selected by their indices
indices	indices of the plots to which the x axis label is to be given. By default the x axis label is given to all plots
...	additional arguments to set_labels, such as action

stackplot *Plots on Top of Each Other*

Description

Plots on Top of Each Other

Usage

```
stackplot(
  functionCalls,
  heights.cm = 5,
  margins.top.cm = 0,
  margins.bottom.cm = 0,
  margins.left.cm = 3,
  margins.right.cm = 1,
  envir = parent.frame()
)
```

Arguments

functionCalls	list of expressions each of which is expected to create a plot when being evaluated with eval
heights.cm	heights of plots in cm
margins.top.cm	top margins of plots in cm
margins.bottom.cm	bottom margins of plots in cm
margins.left.cm	left margins of plots in cm
margins.right.cm	right margins of plots in cm
envir	environment in which the expressions given in functionCalls are to be evaluated. Default: parent.frame()

stackplotLayout *Create Layout for "stackplot"*

Description

Create Layout for "stackplot"

Usage

```
stackplotLayout(heights.cm, margins.top.cm, margins.bottom.cm, show = FALSE)
```

Arguments

`heights.cm`	heights of plots in cm
`margins.top.cm`	top margins of plots in cm
`margins.bottom.cm`	bottom margins of plots in cm
`show`	if `TRUE` the layout is shown in the form of a plot with the borders between the different plot sections being indicated

---

`to_element_themes` *to element themes*

Description

to element themes

Usage

```
to_element_themes(names, element)
```

Arguments

`names`	name of the argument given to `theme`
`element`	value of the argument given to `theme`

`userCoordinatesToLimits` *userCoordinatesToLimits*

Description

back-calculate xlim and ylim from user coordinates of plot region

Usage

```
userCoordinatesToLimits(userCoordinates)
```

Arguments

`userCoordinates`	list with elements `width`, `height`, `left`, `bottom`, `right`, `top`, as returned by `getPlotRegionSizeInUserCoords`

Value

list with elements *xlim* and *ylim*, each of which is a numeric vector of length two.

`userWidthAndHeightToCm` *Convert Lenghts in User Coordinates to Lengths in Centimetres*

Description

Convert Lenghts in User Coordinates to Lengths in Centimetres

Usage

```
userWidthAndHeightToCm(width.user, height.user)
```

Arguments

`width.user`	width in user coordinates
`height.user`	height in user coordinates

Index