Package: ospsuite.plots (via r-universe)

July 6, 2026

Type Package

Title Library for standardized graphs

Version 1.2.0

Description This library provides a standardized approach to creating graphs and tables in the ospsuite context. It is based on ggplot2.

URL https://www.open-systems-pharmacology.org/OSPSuite.Plots/

License GPL-2 | file LICENSE

Language en-US

Depends R (>= 4.1.0), ggplot2 (>= 4.0)

Imports checkmate, cowplot, data.table, dplyr, fitdistrplus, fs, ggh4x, ggnewscale, ospsuite.utils (>= 1.4.0), R6, rlang, scales, tidyr

Suggests devtools (>= 2.4.5), ggpubr, ggsci, knitr, lintr, png, rex, rmarkdown, spelling, svglite, testthat (>= 3.0.3), vdiffr (>= 1.0.0), withr

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Vignette inst/doc/ospsuite-plots.Rmd inst/doc/plot-time-profile.Rmd inst/doc/histogram.Rmd inst/doc/box-whisker-plots.Rmd inst/doc/goodness-of-fit.Rmd inst/doc/forest-plots.Rmd inst/doc/range-plot-visualization.Rmd

Config/testthat/edition 3

Collate 'messages.R' 'utilities-defaults.R' 'utilities_enum.R' 'add_Layer.R' 'ggplotWithWatermark.R' 'geom-errorbar-osp.R' 'CombinedPlot.R' 'data.R' 'MappedData.R' 'MappedDataBoxplot.R' 'MappedDataTimeProfile.R' 'MappedDataRangePlot.R' 'ospsuite.plots-package.R' 'plotBoxWhisker.R' 'plotHistogram.R' 'plotQQ.R' 'plotTimeProfile.R' 'plotYVsX.R' 'plotForest.R' 'plotRangeDistribution.R' 'utilities.R' 'utilities_export.R' 'utilities_shapes.R'

LazyData true

Remotes ospsuite.utils=Open-Systems-Pharmacology/OSPSuite.RUtils@*release

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository https://open-systems-pharmacology.r-universe.dev

Date/Publication 2026-06-01 12:18:06 UTC

RemoteUrl https://github.com/Open-Systems-Pharmacology/OSPSuite.Plots

RemoteRef v1.2.0

RemoteSha 23c7e6519f6e625687367259ac54d063217cf822

Contents

`addLLOQLayer` *Add LLOQ Layer with LLOQ Lines*

Description

Add a layer for LLOQ lines to a `ggplot` object.

Usage

```
addLLOQLayer(
  plotObject,
  mappedData,
  layerToCall,
  useLinetypeAsAttribute,
  geomLLOQAttributes
)
```

Arguments

`plotObject`	A `ggplot` object on which to add the plot layer.
`mappedData`	A `MappedData` object with LLOQ data.
`layerToCall`	A function representing the ggplot2 geom layer.

useLinetypeAsAttribute
: A boolean indicating whether to set the line type as an attribute (TRUE) or not (FALSE); if TRUE, no legend is created.

geomLLOQAttributes
: Additional attributes for the LLOQ layer.

Value

The updated ggplot object.

---

addWatermark *Add a watermark to a ggplot object*

Description

This function adds a customizable watermark to a ggplot object. The watermark can be configured with various options such as position, angle, font size, color, and transparency.

Usage

```
addWatermark(plotObject)
```

Arguments

plotObject	A ggplot object to which the watermark will be added.

Value

A ggplot object with a watermark drawn on it. The watermark is displayed according to the specified options.

See Also

Other watermark: ggplotWithWatermark(), plot.ggWatermark(), print.ggWatermark()

Examples

```
## Not run:
# Example usage (watermark enabled by default)
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
p_with_watermark <- addWatermark(p)
print(p_with_watermark)

# Example of customizing the watermark
setOspsuite.plots.option(optionKey = OptionKeys$watermarkLabel, value = "Custom Watermark")
watermarkFormat <- getOspsuite.plots.option(optionKey = OptionKeys$watermarkFormat)
watermarkFormat$x <- 0.5 # Centered horizontally
watermarkFormat$y <- 0.5 # Centered vertically
```

```
watermarkFormat$angle <- 45 # Rotated 45 degrees
watermarkFormat$fontsize <- 6 # Font size 6
watermarkFormat$color <- "blue" # Blue color
watermarkFormat$alpha <- 0.5 # 50% transparency
setOspsuite.plots.option(optionKey = OptionKeys$watermarkFormat, value = watermarkFormat)

# Create plot with customized watermark
p_custom <- addWatermark(p)
print(p_custom)

## End(Not run)
```

`addXScale` *add X-scale*

Description

add X-scale

Usage

```
addXScale(plotObject, xScale, xScaleArgs = list())
```

Arguments

`plotObject`	A `ggplot` object on which to add the scale.
`xScale`	The x-axis scale type. Available is 'linear', 'log', 'discrete'
`xScaleArgs`	A list of arguments for the x-axis scale.

Value

The updated `ggplot` object

`addXYScale` *Add X and Y Scale*

Description

Add X and Y scales to a `ggplot` object.

Usage

```
addXYScale(
  plotObject,
  xScale = NULL,
  xScaleArgs = list(),
  yScale = NULL,
  yScaleArgs = list(),
  secAxis = waiver()
)
```

Arguments

plotObject	A ggplot object on which to add the scale.
xScale	The x-axis scale type. Available is 'linear', 'log', 'discrete'
xScaleArgs	A list of arguments for the x-axis scale.
yScale	The y-axis scale type. Available is 'linear', 'log'
yScaleArgs	A list of arguments for the y-axis scale.
secAxis	Secondary axis arguments for scale_y functions.

Value

The updated ggplot object.

---

addYScale *add y-scale*

Description

add y-scale

Usage

```
addYScale(plotObject, yScale, yScaleArgs = list(), secAxis = waiver())
```

Arguments

plotObject	A ggplot object on which to add the scale.
yScale	The y-axis scale type. Available is 'linear', 'log'
yScaleArgs	A list of arguments for the y-axis scale.
secAxis	Secondary axis arguments for scale_y functions.

Value

The updated ggplot object

AxisScales *enumeration keys for OSPSuite.plots scaling options for axis scalings*

Description

enumeration keys for OSPSuite.plots scaling options for axis scalings

Usage

```
AxisScales
```

BINNINGMODE *enumeration keys for mode of Binning*

Description

enumeration keys for mode of Binning

Usage

```
BINNINGMODE
```

colorMaps *Color maps*

Description

List with some color maps for `Theme` object.

The `ospDefault` color map is based on colors in `default_igv` qualitative color palette from `{ggsci}` package.

Usage

```
colorMaps
```

See Also

Other setDefault functions: `getDefaultGeomAttributes()`, `getDefaultOptions()`, `getOspsuite.plots.option()`, `resetDefaultColorMapDistinct()`, `resetDefaultTheme()`, `resetDefaults()`, `setDefaultColorMapDistinct()`, `setDefaultTheme()`, `setDefaults()`, `setOspsuite.plots.option()`

CombinedPlot *CombinedPlot*

Description

This class represents a combined plot object that includes a plot and an optional table. It provides methods to get and set the plot and table objects, as well as to print the combined output.

Active bindings

`plotObject` A ggplot object representing the main plot.

`tableObject` A ggplot object representing the table.

`relWidths` A numeric vector of length 2 specifying the relative widths of the plot and table.

Methods

Public methods:

- `CombinedPlot$new()`
- `CombinedPlot$combined()`
- `CombinedPlot$print()`
- `CombinedPlot$clone()`

`CombinedPlot$new()`:

Usage:

```
CombinedPlot$new(plotObject = ggplot(), tableObject = NULL)
```

Arguments:

`plotObject` A ggplot object for the main plot. Combine the combined plot and table
This method combines the plot and table into a single output and displays it.

`tableObject` A ggplot object for the table.

`CombinedPlot$combined()`:

Usage:

```
CombinedPlot$combined()
```

Returns: A ggplot object representing the combined plot and table Print the combined plot and table
This method overrides the default print function to display the combined output.

`CombinedPlot$print()`:

Usage:

```
CombinedPlot$print()
```

Returns: Invisibly returns the combined ggplot object

`CombinedPlot$clone()`: The objects of this class are cloneable with this method.

Usage:

```
CombinedPlot$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:

# Create a new CombinedPlot instance
combinedPlotInstance <- CombinedPlot$new(plotObject = myPlotObject, tableObject = myTableObject)

# Print the combined plot and table
print(combinedPlotInstance)
# or simply
combinedPlotInstance

## End(Not run)
```

`constructLabelWithUnit` *Construct a Label with Unit*

Description

This function constructs a label by appending a unit in square brackets if both the label and unit are provided. If the unit is empty or NULL, only the label is returned.

Usage

```
constructLabelWithUnit(label, unit)
```

Arguments

`label`	A character string representing the label. It should not be NULL.
`unit`	A character string representing the unit. It can be NULL or an empty string.

Value

A character string that combines the label and the unit, formatted as "label unit", or just the label if the unit is empty or NULL.

Examples

```
constructLabelWithUnit("Temperature", "Celsius") # Returns "Temperature [Celsius]"
constructLabelWithUnit("Length", "") # Returns "Length"
constructLabelWithUnit(NULL, "kg") # Returns NULL
```

exampleDataCovariates *Example Covariates Data*

Description

A data frame containing example covariate data used in the vignette for demonstrating histogram plotting functionality with different grouping variables.

Usage

```
data(exampleDataCovariates)
```

Format

A data frame with covariate information typically used in population pharmacokinetic/pharmacodynamic analyses.

Source

Generated for package demonstration purposes.

References

```
vignette("Histogram Plots", package = "ospsuite.plots")
```

exampleDataTimeProfile *Example Time Profile Data*

Description

A data frame containing example pharmacokinetic time profile data used in the vignette for demonstrating time profile plotting functionality.

Usage

```
data(exampleDataTimeProfile)
```

Format

A data frame with time-series pharmacokinetic data containing variables typically used for concentration-time profiles in PBPK modeling.

Source

Generated for package demonstration purposes.

References

```
vignette("Time Profile Plots", package = "ospsuite.plots")
```

`exportPlot` *Export a ggplot object to a file*

Description

This function exports a ggplot object to a specified file with customizable options.

Usage

```
exportPlot(
  plotObject,
  filepath,
  filename,
  width = NULL,
  height = NULL,
  device = NULL,
  ...
)
```

Arguments

`plotObject`	A ggplot object to be exported.
`filepath`	A character string specifying the directory to save the plot.
`filename`	A character string specifying the name of the file (without path).
`width`	A numeric value specifying the width of the plot. If NULL, the default option is used.
`height`	A numeric value specifying the height of the plot. If NULL, it is calculated based on the plot dimensions.
`device`	Export device, if NULL (default) the device set by ospsuite.plots.exportDevice is used.
`...`	Additional arguments passed to `ggsave`.

Details

The height of the plot is calculated if it is not provided by the user. The calculation takes into account:

- The aspect ratio of the plot, which is derived from the theme settings.
- The number of rows and columns in the plot layout.
- The dimensions of plot components such as axes, legends, and margins. The function ensures that the height is adjusted to maintain the correct aspect ratio based on the specified width.

Options available for plot export with default values:

- `ospsuite.plots.exportWidth`: Width of the exported plot (default = 16).
- `ospsuite.plots.exportUnits`: Units of the exported plot (default = "cm").
- `ospsuite.plots.exportDevice`: File format of the exported plot (default = "png").
- `ospsuite.plots.exportDpi`: Resolution of the exported plot (default = 300).

For more details and examples see the vignettes:

- `vignette("ospsuite.plots", package = "ospsuite.plots")`

Value

NULL, the function saves the plot to the specified file.

Examples

```
## Not run:
# Basic usage
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
exportPlot(
  plotObject = p,
  filepath = tempdir(),
  filename = "my_plot.png"
)

# Export with custom dimensions and device
exportPlot(
  plotObject = p,
  filepath = "./output",
  filename = "scatter_plot",
  width = 12,
  height = 8,
  device = "pdf"
)

# Export with special characters in filename (will be cleaned)
exportPlot(
  plotObject = p,
  filepath = tempdir(),
  filename = "concentration in µg/L: results"
)

## End(Not run)
```

geom_errorbar_osp *OSP Errorbar Layer*

Description

A geom that renders error bars with cap `width` specified in **mm** units, keeping it visually consistent with the `linewidth` aesthetic, which is also expressed in **mm**. Unlike `ggplot2::geom_errorbar()`, the cap width is independent of the data coordinate range or axis scale.

Vertical orientation (`aes(x, ymin, ymax)`) is used by default. Pass `orientation = "x"` for horizontal error bars (`aes(y, xmin, xmax)`).

Usage

```
geom_errorbar_osp(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  orientation = NA,
  width = 2,
  lineend = "butt",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`orientation` Orientation of the layer. `"x"` produces horizontal error bars (range along the x-axis). Any other value, including `NA` (default) and `"y"`, produces vertical error bars (range along the y-axis).

`width` Width of the error bar caps in mm units. Default: `2`.

`lineend` Line end style (round, butt, square).

`na.rm` Missing values in the range aesthetics are dropped (the affected cap is simply not drawn) for both `TRUE` and `FALSE`. No warning is emitted in either case. The

argument is kept for consistency with the ggplot2::geom_errorbar() interface.

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

inherit.aes — If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer that can be added to a plot.

See Also

Other layers: GeomErrorbarOsp, GeomPointOsp, geom_point_osp(), stat_qq_osp()

Examples

```
library(ggplot2)
df <- data.frame(
  x    = 1:3,
  y    = c(1, 2, 3),
  ymin = c(0.5, 1.5, 2.5),
  ymax = c(1.5, 2.5, 3.5)
)
ggplot(df, aes(x, y, ymin = ymin, ymax = ymax)) +
  geom_errorbar_osp(width = 2, linewidth = 0.8)
```

---

geom_point_osp *OSP Point Layer*

Description

A geom that renders OSP shapes using grid primitives. Uses shape names from ospShapeNames. Automatically applies scale_shape_osp() when added to a plot (unless a shape scale is already present).

Usage

```
geom_point_osp(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
```

```
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the

available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`na.rm` If `FALSE` (default), missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

Value

A ggplot2 layer that can be added to a plot.

See Also

Other layers: `GeomErrorbarOsp`, `GeomPointOsp`, `geom_errorbar_osp()`, `stat_qq_osp()`

Examples

```
library(ggplot2)
df <- data.frame(x = 1:5, y = 1:5, shape = ospShapeNames[1:5])
ggplot(df, aes(x, y, shape = shape)) +
  geom_point_osp(size = 4) +
  scale_shape_osp_identity()
```

GeomErrorbarOsp *GeomErrorbarOsp*

Description

ggproto object for OSP error bars with cap width in mm units. Use `geom_errorbar_osp()` to add this geom to a ggplot.

See Also

Other layers: `GeomPointOsp`, `geom_errorbar_osp()`, `geom_point_osp()`, `stat_qq_osp()`

GeomPointOsp *GeomPointOsp*

Description

ggproto object for OSP point shapes.

See Also

Other layers: `GeomErrorbarOsp`, `geom_errorbar_osp()`, `geom_point_osp()`, `stat_qq_osp()`

getDefaultGeomAttributes *get the defaults for the geom attributes used as defaults in plot functions see* `vignette("ospsuite.plots", package = "ospsuite.plots")` *how to change defaults*

Description

get the defaults for the geom attributes used as defaults in plot functions see `vignette("ospsuite.plots", package = "ospsuite.plots")` how to change defaults

Usage

```
getDefaultGeomAttributes(geom)
```

Arguments

`geom`	type of geom to return attributes

Value

list with default attributes

See Also

Other setDefault functions: colorMaps, getDefaultOptions(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

---

getDefaultOptions *get list of default options*

Description

get list of default options

Usage

```
getDefaultOptions()
```

Value

names list with default options

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

---

getFoldDistanceList *creates a list with fold Distances*

Description

this list is used as input for plotRatioVsCov, plotPredVsObs

Usage

```
getFoldDistanceList(folds = c(1.5, 2), includeIdentity = TRUE)
```

Arguments

folds	of folds e.g. c(1.5,2) must be >1
includeIdentity	A boolean, if TRUE (default) line of identity is added

Value

named list with fold distances

getOspsuite.plots.option *returns an option value for a option defined by the package OSPSuite.plots*

Description

returns an option value for a option defined by the package OSPSuite.plots

Usage

```
getOspsuite.plots.option(optionKey)
```

Arguments

optionKey	identifier of option

Value

option value

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), resetDefaultColorMapDistinct(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

Examples

```
## Not run:
getOspsuite.plots.option(optionKey = OptionKeys$watermarkEnabled)

## End(Not run)
```

ggplotWithWatermark *Create a ggplot with an optional watermark*

Description

This function creates a ggplot object and adds a watermark if the watermark option is enabled. The watermark can be customized with various options such as position, angle, font size, color, and transparency.

Usage

```
ggplotWithWatermark(...)
```

Arguments

...	Arguments to be passed to `ggplot()`, such as data and aesthetics. This allows for flexibility in creating different types of plots.

Details

If the watermark feature is enabled, the resulting ggplot object will include a watermark overlay when printed. The watermark's properties are determined by options set in the Ospsuite plotting configuration.

The following options can be used to customize the watermark:

- `watermarkLabel`: Text to be displayed as the watermark.
- `watermarkFormat`: A list with the following entries:
  - `x`: The x-coordinate for the watermark's position on the plot.
  - `y`: The y-coordinate for the watermark's position on the plot.
  - `angle`: The angle at which the watermark text is displayed (in degrees).
  - `fontsize`: The size of the font for the watermark text.
  - `color`: The color of the watermark text, specified in a valid color format (e.g., "red", "#FF0000").
  - `alpha`: The transparency level of the watermark text, ranging from 0 (completely transparent) to 1 (completely opaque).

Value

A ggplot object, which may have the class "ggWatermark" if the watermark is enabled. The object can be printed or further modified as needed.

See Also

Other watermark: `addWatermark()`, `plot.ggWatermark()`, `print.ggWatermark()`

Examples

```
# Example usage with watermark enabled (watermark is enabled by default)
plotWithWatermark <- ggplotWithWatermark(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point()
print(plotWithWatermark)

# Example usage with watermark disabled
setOspsuite.plots.option(optionKey = OptionKeys$watermarkEnabled, value = FALSE)
plotWithoutWatermark <- ggplotWithWatermark(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point()
print(plotWithoutWatermark)
# Reset options
setOspsuite.plots.option(optionKey = OptionKeys$watermarkEnabled, value = TRUE)

# Example usage with customized watermark
setOspsuite.plots.option(optionKey = OptionKeys$watermarkLabel, value = "Custom Label")
watermarkFormat <- getOspsuite.plots.option(optionKey = OptionKeys$watermarkFormat)
```

```
watermarkFormat$color <- "red"
setOspsuite.plots.option(optionKey = OptionKeys$watermarkFormat, value = watermarkFormat)
plotWithCustomizedWatermark <- ggplotWithWatermark(data = mtcars, aes(x = wt, y = mpg)) +
  geom_point()
print(plotWithCustomizedWatermark)
# Reset options
setOspsuite.plots.option(
  optionKey = OptionKeys$watermarkFormat,
  value = getDefaultOptions()[[OptionKeys$watermarkFormat]]
)
setOspsuite.plots.option(
  optionKey = OptionKeys$watermarkLabel,
  value = getDefaultOptions()[[OptionKeys$watermarkLabel]]
)
```

`initializePlot` *Initialize Plot*

Description

Initialize a `ggplot` object with a watermark and set its labels by metaData.

Usage

```
initializePlot(mappedData = NULL, setMapping = TRUE)
```

Arguments

`mappedData`	A `MappedData` object.
`setMapping`	A boolean indicating if TRUE (default) mapping is passed to ggplot; otherwise, mapping will be used only to create labels.

Value

A `ggplot` object.

`MappedData` *MappedData*

Description

R6 class for mapping variables to `data`

Public fields

`data` data.frame used for mapping

`mapping` list of aesthetic mappings

`dimensions` list with dimensions of mapping

`units` list with dimensions of mapping

`columnClasses` list with class of mapped columns

`xlimits` double vector limits of primary y axis

`ylimits` double vector limits of primary y axis

Active bindings

`hasLLOQMatch` `boolean` if TRUE data has matched lloq data

`dataForPlot` returns data used for plotting, may be adjusted in child classes (e.g. 2 axis in MappedDataTimeProfile) check if aesthetic is available in data returns data column for aesthetic adds and update mapping deletes data where mdv is 1

adds new column `isLLOQ.i` and updates boolean `LLOQMatch` adds new columns `ymin` and `ymax` if required copy aesthetics `groupby`, but only if not explicit set converts Integer columns, which are no factors to double factorize column for group to factor

Methods

Public methods:

- `MappedData$new()`
- `MappedData$getAestheticsForGeom()`
- `MappedData$addMetaData()`
- `MappedData$updateScaleArgumentsForTimeUnit()`
- `MappedData$clone()`

`MappedData$new()`: Create a new `MappedData` object

Usage:

```
MappedData$new(
  data,
  mapping,
  xScale,
  yScale,
  groupAesthetics = NULL,
  groupOrder = NULL,
  direction = "y",
  isObserved = TRUE,
  xlimits = NULL,
  ylimits = NULL
)
```

Arguments:

`data` data.frame used for mapping

`mapping` list of aesthetic mappings

`xScale` scale of x-axis either 'linear' or 'log'

`yScale` scale of y-axis either 'linear' or 'log'

`groupAesthetics` vector of aesthetics, which are used for columns mapped with `groupby`

`groupOrder` labels and order for group aesthetic

`direction` direction of plot either "x" or "y"

`isObserved` A `boolean` if TRUE mappings mdv, lloq

`xlimits` limits for x-axis (may be NULL)

`ylimits` limits for y-axis (may be NULL)

Returns: A new `MappedData` object filter possible aesthetics for a geom, check if mandatory are available

`MappedData$getAestheticsForGeom()`:

Usage:

```
MappedData$getAestheticsForGeom(geom, geomAttributes)
```

Arguments:

`geom` type of geometric object

`geomAttributes` additionally arguments for geom layer, will overwrite aesthetics

Returns: list of accepted mappings adds list with dimension, units and column classes

`MappedData$addMetaData()`:

Usage:

```
MappedData$addMetaData(metaData)
```

Arguments:

`metaData` A named list of information about `data` such as the `dimension` and `unit` of its variables.

Returns: updated `MappedData` object check if unit of scale direction is time and sets the breaks accordingly

`MappedData$updateScaleArgumentsForTimeUnit()`:

Usage:

```
MappedData$updateScaleArgumentsForTimeUnit(scaleArgs, scaleDirection = "x")
```

Arguments:

`scaleArgs` additional arguments passed on to scale function

`scaleDirection` direction of axis either 'x' or 'y'

Returns: `scaleArgs` with adjusted break function

`MappedData$clone()`: The objects of this class are cloneable with this method.

Usage:

```
MappedData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other MappedData classes: `MappedDataBoxplot`, `MappedDataRangeDistribution`, `MappedDataTimeProfile`

MappedDataBoxplot *MappedDataBoxplot*

Description

R6 class for mapping variable to data for boxplot visualizations. This class extends MappedData to provide specialized mapping functionality for box-and-whisker plots, including handling of discrete and continuous x-axis scales and automatic grouping logic.

Super class

MappedData -> MappedDataBoxplot

Public fields

`xScale` scale of x axis

`xScaleArgs` arguments for scale of x axis

`hasXmapping` boolean, if TRUE x is mapped

Active bindings

`boxwhiskerMapping` mapping for box whisker plot

Methods

Public methods:

- MappedDataBoxplot$new()
- MappedDataBoxplot$doAdjustmentsWithMetaData()
- MappedDataBoxplot$clone()

`MappedDataBoxplot$new()`: Create a new `MappedDataBoxplot` object

Usage:

```
MappedDataBoxplot$new(
  data,
  mapping,
  groupAesthetics = NULL,
  direction = "y",
  isObserved = TRUE,
  xlimits = NULL,
  ylimits = NULL,
  xScale = AxisScales$linear,
  yScale = AxisScales$linear
)
```

Arguments:

`data` data.frame used for mapping

`mapping` list of aesthetic mappings

`groupAesthetics` vector of aesthetics, which are used for columns mapped with aesthetic groupby

`direction` direction of plot either "x" or "y"

`isObserved` A boolean if TRUE mappings mdv, lloq, error and error_relative are evaluated

`xlimits` limits for x-axis (may be NULL)

`ylimits` limits for y-axis (may be NULL)

`xScale` scale of x-axis either 'linear' or 'log'

`yScale` scale of y-axis either 'linear' or 'log'

Returns: `MappedDataBoxplot` class object use Metadata to adjust binning of x-axis, and group aesthetic

`MappedDataBoxplot$doAdjustmentsWithMetaData()`:

Usage:

```
MappedDataBoxplot$doAdjustmentsWithMetaData(
  originalmapping,
  xScale,
  xScaleArgs
)
```

Arguments:

`originalmapping` mapping provided by user

`xScale` either 'linear','log', 'discrete' or 'auto' (default) auto select linear for continuous data and discrete for categorical data

`xScaleArgs` list of arguments passed to `ggplot2::scale_x_continuous()`, `ggplot2::scale_x_log10()` or `ggplot2::scale_x_discrete()`

Returns: adjusted `MappedDataBoxplot` class object

`MappedDataBoxplot$clone()`: The objects of this class are cloneable with this method.

Usage:

```
MappedDataBoxplot$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other MappedData classes: `MappedData`, `MappedDataRangeDistribution`, `MappedDataTimeProfile`

Examples

```
## Not run:
# Create boxplot mapping with continuous x variable
boxplotData <- MappedDataBoxplot$new(
  data = myDataFrame,
  mapping = aes(x = dose, y = concentration),
  xScale = "linear"
```

```
)

# Create boxplot mapping with categorical x variable
boxplotData <- MappedDataBoxplot$new(
  data = myDataFrame,
  mapping = aes(x = treatment_group, y = response),
  xScale = "discrete"
)

## End(Not run)
```

MappedDataRangeDistribution
object to map data for rangeplots

Description

R6 class for mapping variable to data

Super class

MappedData -> MappedDataRangeDistribution

Public fields

xScale scale of x axis

Active bindings

border borders of the binning.

Methods

Public methods:

- MappedDataRangeDistribution$new()
- MappedDataRangeDistribution$setBins()
- MappedDataRangeDistribution$setBorderDataTable()
- MappedDataRangeDistribution$setXMapping()
- MappedDataRangeDistribution$clone()

MappedDataRangeDistribution$new(): Create a new MappedDataRangeDistribution object

Usage:

```
MappedDataRangeDistribution$new(
  data,
  mapping,
  groupAesthetics = NULL,
  direction = "y",
```

```
  isObserved = TRUE,
  xlimits = NULL,
  ylimits = NULL,
  xScale = "linear",
  yScale = "linear",
  modeOfBinning = NA,
  numberOfBins = NA,
  breaks = NA
)
```

Arguments:

`data` data.frame used for mapping

`mapping` list of aesthetic mappings

`groupAesthetics` vector of aesthetics, which are used for columns mapped with aesthetic groupby

`direction` direction of plot either "x" or "y"

`isObserved` A `boolean` if TRUE mappings mdv, lloq, error and error_relative are evaluated

`xlimits` limits for x-axis (may be NULL)

`ylimits` limits for y-axis (may be NULL)

`xScale` scale of x-axis either 'linear' or 'log'

`yScale` scale of y-axis either 'linear' or 'log'

`modeOfBinning` method of binning (e.g., 'breaks', 'number', 'interval')

`numberOfBins` number of bins to use for binning

`breaks` breaks for binning if `modeOfBinning` is 'breaks'

Returns: `MappedDataRangeDistribution` class object Set binning columns

`MappedDataRangeDistribution$setBins()`: This method sets the bins for the data based on the specified mode of binning.

Usage:

```
MappedDataRangeDistribution$setBins()
```

Returns: The object itself (invisible) Create a data table with bin border information

`MappedDataRangeDistribution$setBorderDataTable()`: This method sets up a data table containing border information for the bins. Set x mapping for the plot

Usage:

```
MappedDataRangeDistribution$setBorderDataTable(identifier = "IndividualId")
```

Arguments:

`identifier` Identifier for the data table (default is 'IndividualId')

`MappedDataRangeDistribution$setXMapping()`: This method sets the x mapping for the plot based on the specified parameters.

Usage:

```
MappedDataRangeDistribution$setXMapping(asStepPlot)
```

Arguments:

`asStepPlot` Logical indicating if the plot should be a step plot.

`MappedDataRangeDistribution$clone()`: The objects of this class are cloneable with this method.

Usage:

```
MappedDataRangeDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other MappedData classes: `MappedData`, `MappedDataBoxplot`, `MappedDataTimeProfile`

`MappedDataTimeProfile` *MappedDataTimeProfile*

Description

R6 class for mapping variable to data for time profile visualizations. This class extends MappedData to provide specialized functionality for time-series plots, including support for secondary y-axes, dual scaling, and time-specific axis handling.

Details

This class is specifically designed for pharmacokinetic time profile plots where data may need to be displayed on dual y-axes with different scales (linear/log). It handles complex scenarios like mapping simulated and observed data with different scaling requirements.

Super class

`MappedData` -> `MappedDataTimeProfile`

Public fields

`y2limits` double vector limits of secondary y axis

Active bindings

`requireDualAxis` boolean, If TRUE secondary axis is required

`listOfGroups` character vector of groupings

`secAxis` sec_axis() object

`dataForPlot` scaled data used for plotting adjust limits

Methods

Public methods:

- MappedDataTimeProfile$new()
- MappedDataTimeProfile$scaleDataForSecondaryAxis()
- MappedDataTimeProfile$clone()

`MappedDataTimeProfile$new()`: Create a new `MappedDataTimeProfile` object

Usage:

```
MappedDataTimeProfile$new(
  data,
  mapping,
  groupAesthetics = NULL,
  groupOrder = NULL,
  direction = "y",
  isObserved = TRUE,
  xlimits = NULL,
  ylimits = NULL,
  xScale = AxisScales$linear,
  scaleOfPrimaryAxis = AxisScales$linear,
  scaleOfSecondaryAxis = AxisScales$linear,
  y2limits = NULL
)
```

Arguments:

`data` data.frame used for mapping

`mapping` list of aesthetic mappings

`groupAesthetics` vector of aesthetics, which are used for columns mapped with aesthetic `groupby` , use of group aesthetics triggers second axis after simulation layers

`groupOrder` labels and order for group aesthetic

`direction` direction of plot either "x" or "y"

`isObserved` A `boolean` if TRUE mappings mdv, lloq are evaluated

`xlimits` limits for x-axis (may be NULL)

`ylimits` limits for primary axis (may be NULL)

`xScale` = scale of x-axis

`scaleOfPrimaryAxis` scale of direction, either "linear" or "log"

`scaleOfSecondaryAxis` either 'linear' or 'log'

`y2limits` limits for secondary axis (may be NULL)

Returns: A new `MappedDataTimeProfile` object Scale data for secondary axis and update secAxis transformation
This method handles the complex logic of scaling data between primary and secondary axes with different scale types (linear/log combinations).

`MappedDataTimeProfile$scaleDataForSecondaryAxis()`:

Usage:

```
MappedDataTimeProfile$scaleDataForSecondaryAxis(
  ylimits = NULL,
  y2limits = NULL,
  y2ScaleArgs = list()
)
```

Arguments:

`ylimits` limits for primary axis (may be NULL)

`y2limits` limits for secondary axis (may be NULL)

`y2ScaleArgs` arguments for secondary axis

Returns: updated MappedDataTimeProfile boolean for secondary axis

`MappedDataTimeProfile$clone()`: The objects of this class are cloneable with this method.

Usage:

```
MappedDataTimeProfile$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other MappedData classes: `MappedData`, `MappedDataBoxplot`, `MappedDataRangeDistribution`

Examples

```
## Not run:
# Create time profile mapping with secondary axis
timeData <- MappedDataTimeProfile$new(
  data = myDataFrame,
  mapping = aes(x = time, y = concentration, y2axis = fraction_unbound),
  scaleOfPrimaryAxis = "linear",
  scaleOfSecondaryAxis = "log"
)

# Time profile with grouping aesthetics
timeData <- MappedDataTimeProfile$new(
  data = myDataFrame,
  mapping = aes(x = time, y = concentration, color = compound),
  groupAesthetics = c("color", "linetype")
)

## End(Not run)
```

metaData2DataFrame *converts metaData List to a data frame row names specify properties*

Description

converts metaData List to a data frame row names specify properties

Usage

```
metaData2DataFrame(metaData)
```

Arguments

metaData	A named list of information about the data such as the dimension and unit of its variables.

Value

metaData as data.frame

OptionKeys *enumeration keys for OSPSuite.plots options*

Description

enumeration keys for OSPSuite.plots options

Usage

```
OptionKeys
```

ospShapeNames *OSP Shape Names*

Description

Character vector of all available OSP shape names.

Usage

```
ospShapeNames
```

See Also

Other shapes: Shapes, scale_shape_osp(), scale_shape_osp_identity(), scale_shape_osp_manual()

plot.ggWatermark *Create plot function for ggWatermark.*

Description

This method allows for the ggWatermark object to be plotted using the standard print method.

Usage

```
## S3 method for class 'ggWatermark'
plot(x, ...)
```

Arguments

x	A ggWatermark object to be printed.
...	Additional arguments to be passed to the print method.

See Also

Other watermark: addWatermark(), ggplotWithWatermark(), print.ggWatermark()

plotBoxWhisker *Generate Box-Whisker Plots*

Description

Produces box-and-whisker plots for visualizing the distribution of data. For more details and examples, see the vignettes:

- vignette("Box-Whisker Plots", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")

Usage

```
plotBoxWhisker(
  data,
  mapping,
  metaData = NULL,
  plotObject = NULL,
  percentiles = getOspsuite.plots.option(optionKey = OptionKeys$percentiles),
  yScale = AxisScales$linear,
  yScaleArgs = list(),
  xScale = "auto",
  xScaleArgs = list(),
  statFun = NULL,
```

```
  outliers = FALSE,
  statFunOutlier = NULL,
  geomBoxplotAttributes = getDefaultGeomAttributes("Boxplot"),
  geomPointAttributes = getDefaultGeomAttributes("Boxplot"),
  residualScale = NULL
)
```

Arguments

data	A data.frame containing the data to aggregate.
mapping	A list of aesthetic mappings to use for the plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
plotObject	An optional ggplot object on which to add the plot layers
percentiles	A numeric vector with percentiles used for the box whiskers and boxes, e.g., c(0.05, 0.25, 0.5, 0.75, 0.95). Default defined by ospsuite.plots option.
yScale	either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used
yScaleArgs	list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()
xScale	Either 'linear', 'log', 'discrete', or 'auto' (default). Auto selects linear for continuous data and discrete for categorical data.
xScaleArgs	A list of arguments passed to ggplot2::scale_x_continuous(), ggplot2::scale_x_log10(), or ggplot2::scale_x_discrete().
statFun	(default NULL) A function to calculate whiskers and box ranges, which overwrites the percentiles variable if provided.
outliers	Logical indicating whether outliers should be included in the boxplot. Outliers are flagged when outside the range from the "25th" percentile - 1.5 x IQR to the "75th" percentile + 1.5 x IQR, as suggested by McGill et al.
statFunOutlier	(default NULL) A function to calculate outliers, which overwrites the default calculation if provided.
geomBoxplotAttributes	A list of arguments passed to the geom_boxplot call.
geomPointAttributes	A list of arguments passed to the ggplot2::geom_point call.
residualScale	Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.

Value

A ggplot object representing the box-whisker plot.

References

McGill, R., Tukey, J. W., & Larsen, W. A. (1978). Variations of box plots. The American Statistician, 32(1), 12-16.

See Also

Other plot functions: plotForest(), plotHistogram(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile(), plotYVsX()

Examples

```
## Not run:
# Basic box-whisker plot
plotBoxWhisker(
  data = myData,
  mapping = aes(x = group, y = value)
)

# Box-whisker plot with custom percentiles
plotBoxWhisker(
  data = myData,
  mapping = aes(x = treatment, y = response),
  percentiles = c(0.1, 0.25, 0.5, 0.75, 0.9)
)

# Box-whisker plot with custom stat function
customStatFun <- function(x) {
  return(quantile(x, probs = c(0.05, 0.25, 0.5, 0.75, 0.95), na.rm = TRUE))
}
plotBoxWhisker(
  data = myData,
  mapping = aes(x = dose_group, y = concentration),
  statFun = customStatFun,
  outliers = TRUE
)

## End(Not run)
```

---

plotForest *Create a Forest Plot*

Description

This function generates a forest plot with optional faceting and a corresponding table.

Usage

```
plotForest(
  plotData,
  mapping = aes(y = y, x = x, groupby = dataType),
  xLabel,
  yFacetColumns = NULL,
  xFacetColumn = NULL,
  xScale = c("linear", "log"),
```

```
  xScaleArgs = list(),
  groupAesthetics = c("color", "fill", "shape"),
  tableColumns = c("yValues", "yErrorValues"),
  tableLabels = c("M", "Variance"),
  labelWrapWidth = 10,
  digitsToRound = 2,
  digitsToShow = 2,
  withTable = is.null(xFacetColumn),
  geomPointAttributes = getDefaultGeomAttributes("Point"),
  geomErrorbarAttributes = getDefaultGeomAttributes("Errorbar"),
  facetScales = c("free_y", "free")
)
```

Arguments

plotData	A data.table containing the data to be plotted. Must include columns specified in yFacetColumns, xFacetColumn, tableColumns, and others.
mapping	A ggplot2 mapping object, typically created with ggplot2::aes(), to specify how variables in the data are mapped to visual properties.
xLabel	A string representing the label for the x-axis.
yFacetColumns	A character vector of column names used for faceting on the y-axis. Can be NULL or of length up to 2.
xFacetColumn	A character string specifying the column name for the x-axis facet. Must be of length 1 or NULL.
xScale	A character string indicating the scale type for the x-axis. Options are "linear" or "log".
xScaleArgs	A list of additional arguments for customizing the x-axis scale.
groupAesthetics	A character vector specifying aesthetics for grouping (e.g., color, fill, shape).
tableColumns	A character vector of column names to be included in the table.
tableLabels	A character vector of labels corresponding to tableColumns.
labelWrapWidth	A numeric value specifying the width for label wrapping in facets.
digitsToRound	An integer specifying the number of digits to round in the table.
digitsToShow	An integer specifying the number of digits to display in the table.
withTable	A logical flag indicating whether to include the table in the output. Defaults to TRUE if xFacetColumn is not NULL.
geomPointAttributes	A list of attributes for the point geometry in the plot.
geomErrorbarAttributes	A list of attributes for the error bar geometry in the plot.
facetScales	A character string indicating the scales used for facets. Options are "free_y" or "free".

Value

A combined plot object containing the forest plot and the table (if applicable).

See Also

Other plot functions: plotBoxWhisker(), plotHistogram(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile(), plotYVsX()

---

plotHistogram *Generates Histograms*

Description

Produces histograms with optional distribution fit.

For more details and examples see the vignettes:

- vignette("Histogram Plots", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")
- vignette("Goodness of fit", package = "ospsuite.plots")

Usage

```
plotHistogram(
  data,
  mapping,
  metaData = NULL,
  asBarPlot = NULL,
  geomHistAttributes = getDefaultGeomAttributes("Hist"),
  plotAsFrequency = FALSE,
  xScale = AxisScales$linear,
  xScaleArgs = list(),
  yScale = AxisScales$linear,
  yScaleArgs = list(),
  distribution = "none",
  meanFunction = "auto",
  residualScale = NULL
)
```

Arguments

data	data.frame with simulated data will be displayed as lines with ribbons
mapping	a list of aesthetic mappings to use for plot, additional to {ggplot2} aesthetics, the aesthetics groupby,error,error_relative,lloq, mdv, y2axis are available, see vignettes for more details and examples
metaData	A named list of information about data such as the dimension and unit of its variables.
asBarPlot	A logical indicating if geom_histogram should be used (for continuous data) or geom_bar (for categorical data). If TRUE, the variables distribution, meanFunction, xScale, and xScaleArgs are ignored.

geomHistAttributes
: A list of arguments passed to ggplot2::geom_histogram (or geom_bar if asBarPlot = TRUE).

plotAsFrequency
: A logical indicating if the histogram displays frequency on the y-axis.

xScale
: either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used

xScaleArgs
: list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()

yScale
: either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used

yScaleArgs
: list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()

distribution
: Name of the distribution to fit. Available distributions are those in the stats package (see ?stats::distributions): norm, lnorm, weibull, gamma, etc. Use "none" for no fit (default). Shortcuts: "normal" (same as "norm"), "lognormal" (same as "lnorm").

meanFunction
: Function selection for the display of a vertical line. Options: 'none', 'mean', 'geomean', 'median', 'auto' (default). 'auto' selects 'mean' for normal distribution, 'geomean' for lognormal, 'median' for other distributions, and 'none' when no distribution fit.

residualScale
: Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.

Value

A ggplot object.

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile(), plotYVsX()

---

plotPredVsObs *Generate Predicted vs Observed Plots*

Description

This function is a wrapper for plotYVsX with adjusted input parameters.

The following parameters are fixed and cannot be set:

- observedDataDirection = "x"

For details and examples, see the vignettes:

- vignette("Goodness of fit", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")

Usage

```
plotPredVsObs(
  data = NULL,
  mapping = NULL,
  xyScale = AxisScales$log,
  comparisonLineVector = getFoldDistanceList(c(1.5, 2)),
  asSquarePlot = TRUE,
  ...
)
```

Arguments

data	A data.frame containing the data to plot.
mapping	A list of aesthetic mappings to use for the plot.
xyScale	Either "linear" or "log" scale for the X and Y axes.
comparisonLineVector	A vector defining the comparison lines.
asSquarePlot	A boolean; if true, the plot is returned as a square plot with aspect ratio = 1 and fixed ratios.
...	Arguments passed on to plotYVsX
	geomComparisonLineAttributes A list of arguments passed to ggplot2::hline or ggplot2::abline to display comparison lines.
	geomGuestLineAttributes A list of arguments passed to ggplot2::geom_function to display guest criteria.
	yDisplayAsAbsolute A boolean that defines the direction of comparison lines.
	addRegression A boolean that activates the insertion of a regression line.
	addGuestLimits A boolean that activates the insertion of guest limits.
	deltaGuest Numeric value parameter for the Guest function.
	labelGuestCriteria Label used in the legend for guest criteria (default: "guest criteria").
	observedDataDirection Either "x" or "y", defining the direction of observed data.
	lloqOnBothAxes A boolean; if TRUE, LLOQ lines are drawn on both axes. If FALSE (default), the LLOQ line is drawn for the observed-data axis only. (geom_vline when observedDataDirection = "x", geom_hline when observedDataDirection = "y").
	groupAesthetics A character vector of aesthetic names used for grouping data points when calculating comparison statistics. Data will be grouped by combinations of these aesthetics before computing counts and proportions within comparison lines. Common grouping aesthetics include "colour", "fill", "shape".
	residualScale Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.
	metaData A named list of information about data such as the dimension and unit of its variables.

geomPointAttributes A list with arguments which are passed on to the call ggplot2::geom_point

geomErrorbarAttributes A list with arguments which are passed on to the call geom_errorbar_osp

geomLLOQAttributes A list with arguments which are passed on to the call ggplot2::geom_hline

xScale either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used

xScaleArgs list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()

yScale either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used

yScaleArgs list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()

Value

A ggplot object representing the predicted vs observed plots.

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotHistogram(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile(), plotYVsX()

---

plotQQ *generates residual quantile quantile plot*

Description

For details and examples see the vignettes:

- vignette("Goodness of fit", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")

Usage

```
plotQQ(
  data,
  mapping,
  metaData = NULL,
  xScaleArgs = list(),
  yScaleArgs = list(),
  geomQQAttributes = list(),
  geomQQLineAttributes = geomQQAttributes,
  groupAesthetics = c("colour", "fill", "shape"),
  residualScale = NULL
)
```

Arguments

data	´data.frame‘ with data to plot
mapping	a list of aesthetic mappings to use for plot, additional to {ggplot2} aesthetics, the aesthetics groupby,error,error_relative,lloq, mdv, y2axis are available, see vignettes for more details and examples
metaData	A named list of information about data such as the dimension and unit of its variables.
xScaleArgs	list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()
yScaleArgs	list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()
geomQQAttributes	A list of arguments passed to ggplot2::stat_qq().
geomQQLineAttributes	A list of arguments passed to ggplot2::stat_qq_line().
groupAesthetics	A character vector of aesthetic names used for grouping data points in the Q-Q plot. Common options include "colour", "fill", "shape", "linetype", and "size".
residualScale	Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.

Value

A ggplot object

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotHistogram(), plotPredVsObs(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile(), plotYVsX()

---

plotRangeDistribution *Plot Range Plot*

Description

Creates a range plot using specified data and mapping, allowing for different binning strategies and scaling options. This function provides a flexible way to visualize data distributions over specified ranges with optional statistical summaries.

Usage

```
plotRangeDistribution(
  data,
  mapping,
  metaData = NULL,
  modeOfBinning = BINNINGMODE$number,
```

```
  numberOfBins = 20,
  breaks = NA,
  asStepPlot = FALSE,
  statFun = NULL,
 percentiles = getOspsuite.plots.option(optionKey = OptionKeys$defaultPercentiles),
  yScale = "linear",
  yScaleArgs = list(),
  xScale = "linear",
  xScaleArgs = list(),
  geomRibbonAttributes = getDefaultGeomAttributes("Ribbon"),
  geomLineAttributes = getDefaultGeomAttributes("Line"),
  identifier = "IndividualId"
)
```

Arguments

`data`	A data frame containing the data to be plotted. This data should include the variables specified in the `mapping` argument.
`mapping`	A mapping object (created using `ggplot2::aes()`) that defines how variables in `data` are mapped to aesthetics such as x and y axes, color, fill, etc.
`metaData`	Optional metadata to be added to the plot. This can include additional information relevant to the data being plotted.
`modeOfBinning`	A character string specifying the mode of binning. It determines how the data will be divided into bins. Options include: • `Equal Frequency Binning` • 'Equal Width Binning • `Custom Binning` Default is `BINNINGMODE$number`.
`numberOfBins`	An integer specifying the number of bins to use for equal frequency or width binning. Default is 20.
`breaks`	Optional numeric vector specifying custom breaks for binning when `modeOfBinning` is set to `Custom Binning`. This allows for precise control over how data is grouped.
`asStepPlot`	A logical indicating whether to create a step plot. If TRUE, the plot will display steps between the data points rather than continuous lines. Default is FALSE.
`statFun`	An optional function for statistical summary, which takes a vector of y-values and returns a summary (e.g., quantiles). If NULL, defaults to calculating quantiles based on the specified `percentiles`.
`percentiles`	A numeric vector of percentiles to be used in the statistical summary, which defines the range of values to be displayed on the plot. Default is the 5th, 50th, and 95th percentiles.
`yScale`	A character string specifying the y-axis scale. Options are "linear" or "log". This determines how the y values are displayed on the plot. Default is "linear".
`yScaleArgs`	A list of additional arguments for the y-axis scale, which can be used to customize the appearance and behavior of the y-axis.

xScale	A character string specifying the x-axis scale. Options are "linear" or "log". This determines how the x values are displayed on the plot. Default is "linear".
xScaleArgs	A list of additional arguments for the x-axis scale, which can be used to customize the appearance and behavior of the x-axis.
geomRibbonAttributes	A list of attributes for the ribbon geometry in the plot, allowing customization of the visual appearance, such as colors and transparency.
geomLineAttributes	A list of attributes for the line geometry in the plot, allowing customization of line characteristics such as color, size, and type.
identifier	columnName of individual identifiers, default "IndividualId"

Value

A ggplot object representing the range plot. The returned object can be further customized or rendered using `print()` or similar functions.

See Also

Other plot functions: `plotBoxWhisker()`, `plotForest()`, `plotHistogram()`, `plotPredVsObs()`, `plotQQ()`, `plotRatioVsCov()`, `plotResVsCov()`, `plotTimeProfile()`, `plotYVsX()`

---

plotRatioVsCov *Generate Plots of Ratios vs Covariate*

Description

This function is a wrapper for `plotYVsX` with adjusted input parameters.

The following parameters are fixed and cannot be set:

- `observedDataDirection = "y"`
- `yDisplayAsAbsolute = FALSE`

For details and examples, see the vignettes:

- `vignette("Goodness of fit", package = "ospsuite.plots")`
- `vignette("ospsuite.plots", package = "ospsuite.plots")`

Usage

```
plotRatioVsCov(
  data = NULL,
  mapping = NULL,
  addGuestLimits = FALSE,
  yScale = AxisScales$log,
  xScale = ifelse(addGuestLimits, AxisScales$log, AxisScales$linear),
```

```
  comparisonLineVector = getFoldDistanceList(c(1.5, 2)),
  deltaGuest = 1,
  residualScale = NULL,
  ...
)
```

Arguments

data	A data.frame containing the data to plot.
mapping	A list of aesthetic mappings to use for the plot.
addGuestLimits	A boolean that activates the insertion of guest limits.
yScale	either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used
xScale	either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used
comparisonLineVector	A vector defining the comparison lines.
deltaGuest	Numeric value parameter for the Guest function.
residualScale	Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.
...	Arguments passed on to plotYVsX geomComparisonLineAttributes A list of arguments passed to ggplot2::hline or ggplot2::abline to display comparison lines. geomGuestLineAttributes A list of arguments passed to ggplot2::geom_function to display guest criteria. yDisplayAsAbsolute A boolean that defines the direction of comparison lines. addRegression A boolean that activates the insertion of a regression line. labelGuestCriteria Label used in the legend for guest criteria (default: "guest criteria"). asSquarePlot A boolean; if true, the plot is returned as a square plot with aspect ratio = 1 and fixed ratios. observedDataDirection Either "x" or "y", defining the direction of observed data. lloqOnBothAxes A boolean; if TRUE, LLOQ lines are drawn on both axes. If FALSE (default), the LLOQ line is drawn for the observed-data axis only. (geom_vline when observedDataDirection = "x", geom_hline when observedDataDirection = "y"). groupAesthetics A character vector of aesthetic names used for grouping data points when calculating comparison statistics. Data will be grouped by combinations of these aesthetics before computing counts and proportions within comparison lines. Common grouping aesthetics include "colour", "fill", "shape". metaData A named list of information about data such as the dimension and unit of its variables.

`geomPointAttributes` A `list` with arguments which are passed on to the call `ggplot2::geom_point`

`geomErrorbarAttributes` A `list` with arguments which are passed on to the call `geom_errorbar_osp`

`geomLLOQAttributes` A `list` with arguments which are passed on to the call `ggplot2::geom_hline`

`xScaleArgs` list of arguments passed to `ggplot2::scale_x_continuous()` or `ggplot2::scale_x_log10()`

`yScaleArgs` list of arguments passed to `ggplot2::scale_y_continuous()` or `ggplot2::scale_y_log10()`

Value

A `ggplot` object representing the ratio plots.

See Also

Other plot functions: `plotBoxWhisker()`, `plotForest()`, `plotHistogram()`, `plotPredVsObs()`, `plotQQ()`, `plotRangeDistribution()`, `plotResVsCov()`, `plotTimeProfile()`, `plotYVsX()`

`plotResVsCov` *Generate Residual Plots vs Covariate*

Description

This function is a wrapper for `plotYVsX` with adjusted input parameters.

The following parameters are fixed and cannot be set:

- `observedDataDirection = "y"`
- `yDisplayAsAbsolute = FALSE`
- `addGuestLimits = FALSE` (use `plotRatio()` if needed)

For details and examples, see the vignettes:

- `vignette("Goodness of fit", package = "ospsuite.plots")`
- `vignette("ospsuite.plots", package = "ospsuite.plots")`

Usage

```
plotResVsCov(
  data,
  mapping,
  comparisonLineVector = 0,
  yScale = AxisScales$linear,
  residualScale = NULL,
  ...
)
```

Arguments

data	A data.frame containing the data to plot.
mapping	A list of aesthetic mappings to use for the plot.
comparisonLineVector	A vector defining the comparison lines.
yScale	either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used
residualScale	Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.
...	Arguments passed on to plotYVsX geomComparisonLineAttributes A list of arguments passed to ggplot2::hline or ggplot2::abline to display comparison lines. geomGuestLineAttributes A list of arguments passed to ggplot2::geom_function to display guest criteria. yDisplayAsAbsolute A boolean that defines the direction of comparison lines. addRegression A boolean that activates the insertion of a regression line. addGuestLimits A boolean that activates the insertion of guest limits. deltaGuest Numeric value parameter for the Guest function. labelGuestCriteria Label used in the legend for guest criteria (default: "guest criteria"). asSquarePlot A boolean; if true, the plot is returned as a square plot with aspect ratio = 1 and fixed ratios. observedDataDirection Either "x" or "y", defining the direction of observed data. lloqOnBothAxes A boolean; if TRUE, LLOQ lines are drawn on both axes. If FALSE (default), the LLOQ line is drawn for the observed-data axis only. (geom_vline when observedDataDirection = "x", geom_hline when observedDataDirection = "y"). groupAesthetics A character vector of aesthetic names used for grouping data points when calculating comparison statistics. Data will be grouped by combinations of these aesthetics before computing counts and proportions within comparison lines. Common grouping aesthetics include "colour", "fill", "shape". metaData A named list of information about data such as the dimension and unit of its variables. geomPointAttributes A list with arguments which are passed on to the call ggplot2::geom_point geomErrorbarAttributes A list with arguments which are passed on to the call geom_errorbar_osp geomLLOQAttributes A list with arguments which are passed on to the call ggplot2::geom_hline xScale either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used

xScaleArgs list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()

yScaleArgs list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()

Value

A ggplot object representing the residual plots.

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotHistogram(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotTimeProfile(), plotYVsX()

---

plotTimeProfile *generate time profile plots*

Description

Produces time profiles for simulated and observed data.

For the simulated data a geom_line and a geom_ribbon layer are added For the observed data a geom_point and a geom_errorbar_osp layer are added

For more details and examples see the vignettes:

- vignette("Time Profile Plots", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")

Usage

```
plotTimeProfile(
  data = NULL,
  mapping = NULL,
  observedData = NULL,
  observedMapping = mapping,
  metaData = NULL,
  mapSimulatedAndObserved = NULL,
  xScale = AxisScales$linear,
  xScaleArgs = list(limits = c(0, NA)),
  yScale = AxisScales$linear,
  yScaleArgs = list(),
  y2Scale = AxisScales$linear,
  y2ScaleArgs = list(),
  plotObject = NULL,
  geomLineAttributes = getDefaultGeomAttributes("Line"),
  geomRibbonAttributes = getDefaultGeomAttributes("Ribbon"),
  geomPointAttributes = getDefaultGeomAttributes("Point"),
  geomErrorbarAttributes = getDefaultGeomAttributes("Errorbar"),
```

```
  geomLLOQAttributes = getDefaultGeomAttributes("LLOQ"),
  groupAesthetics = c("colour", "fill", "shape")
)
```

Arguments

data	data.frame with simulated data will be displayed as lines with ribbons
mapping	a list of aesthetic mappings to use for plot, additional to {ggplot2} aesthetics, the aesthetics groupby,error,error_relative,lloq, mdv, y2axis are available, see vignettes for more details and examples
observedData	data.frame with observed data will be displayed as points with error-bars
observedMapping	a list of aesthetic mappings to use for observed data, per default is is set to mapping. So if both data sets have the same mapping, use only mapping, if a different mapping is necessary use mapping and observedMapping
metaData	A named list of information about data such as the dimension and unit of its variables.
mapSimulatedAndObserved	table with columns observed and simulated which maps simulated and observed data use of mapSimulatedAndObserved triggers reset of aesthetic scales after simulation layers
xScale	either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used
xScaleArgs	list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()
yScale	either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used
yScaleArgs	list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()
y2Scale	either 'linear' the secondary axis is displayed linear, or 'log' secondary axis is displayed with log scale
y2ScaleArgs	list of arguments passed to ggplot2::sec_axis(), trans, break are set by code
plotObject	An optional ggplot object on which to add the plot layers
geomLineAttributes	A list with arguments which are passed on to the call ggplot2::geom_line
geomRibbonAttributes	A list with arguments which are passed on to the call ggplot2::geom_ribbon
geomPointAttributes	A list with arguments which are passed on to the call ggplot2::geom_point
geomErrorbarAttributes	A list with arguments which are passed on to the call geom_errorbar_osp
geomLLOQAttributes	A list with arguments which are passed on to the call ggplot2::geom_hline
groupAesthetics	vector of aesthetics, which are used for columns mapped with groupby,

Value

A ggplot object

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotHistogram(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotYVsX()

Examples

```
## Not run:
# Basic time profile plot with simulated data
plotTimeProfile(
  data = simulationData,
  mapping = aes(x = time, y = concentration, color = compound)
)

# Time profile with both simulated and observed data
plotTimeProfile(
  data = simulationData,
  observedData = observedData,
  mapping = aes(x = time, y = concentration, color = treatment),
  observedMapping = aes(x = time, y = concentration, color = treatment)
)

# Time profile with secondary y-axis
plotTimeProfile(
  data = myData,
  mapping = aes(x = time, y = concentration, y2axis = fraction_unbound)
)

## End(Not run)
```

---

plotYVsX *Base Plot for Residuals and Predictions vs Covariates*

Description

This function creates a base plot for plotResVsCov(), plotRatioVsCov(), and plotPredVsObs().

For details and examples, see the vignettes:

- vignette("Goodness of fit", package = "ospsuite.plots")
- vignette("ospsuite.plots", package = "ospsuite.plots")

Usage

```
plotYVsX(
  data,
  mapping,
  metaData = NULL,
  geomPointAttributes = getDefaultGeomAttributes("Point"),
  geomErrorbarAttributes = getDefaultGeomAttributes("Errorbar"),
  geomGuestLineAttributes = getDefaultGeomAttributes("GuestLine"),
  geomComparisonLineAttributes = getDefaultGeomAttributes("ComparisonLine"),
  geomLLOQAttributes = getDefaultGeomAttributes("LLOQ"),
  groupAesthetics = c("colour", "fill", "shape"),
  comparisonLineVector = NULL,
  addRegression = FALSE,
  addGuestLimits = FALSE,
  deltaGuest = 1,
  labelGuestCriteria = "guest criteria",
  residualScale = NULL,
  asSquarePlot = FALSE,
  xScale = AxisScales$linear,
  xScaleArgs = list(),
  yScale = AxisScales$log,
  yScaleArgs = list(),
  observedDataDirection = "y",
  lloqOnBothAxes = FALSE,
  yDisplayAsAbsolute = TRUE
)
```

Arguments

data	A data.frame containing the data to plot.
mapping	A list of aesthetic mappings to use for the plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
geomPointAttributes	A list with arguments which are passed on to the call ggplot2::geom_point
geomErrorbarAttributes	A list with arguments which are passed on to the call geom_errorbar_osp
geomGuestLineAttributes	A list of arguments passed to ggplot2::geom_function to display guest criteria.
geomComparisonLineAttributes	A list of arguments passed to ggplot2::hline or ggplot2::abline to display comparison lines.
geomLLOQAttributes	A list with arguments which are passed on to the call ggplot2::geom_hline
groupAesthetics	A character vector of aesthetic names used for grouping data points when calculating comparison statistics. Data will be grouped by combinations of these

aesthetics before computing counts and proportions within comparison lines. Common grouping aesthetics include "colour", "fill", "shape".

comparisonLineVector
: A vector defining the comparison lines.

addRegression
: A boolean that activates the insertion of a regression line.

addGuestLimits
: A boolean that activates the insertion of guest limits.

deltaGuest
: Numeric value parameter for the Guest function.

labelGuestCriteria
: Label used in the legend for guest criteria (default: "guest criteria").

residualScale
: Deprecated. Retained for backward compatibility only. Non-NULL values trigger a warning and have no effect.

asSquarePlot
: A boolean; if true, the plot is returned as a square plot with aspect ratio = 1 and fixed ratios.

xScale
: either 'linear' then ggplot2::scale_x_continuous() or 'log' then ggplot2::scale_x_log10() is used

xScaleArgs
: list of arguments passed to ggplot2::scale_x_continuous() or ggplot2::scale_x_log10()

yScale
: either 'linear' then ggplot2::scale_y_continuous() or 'log' then ggplot2::scale_y_log10() is used

yScaleArgs
: list of arguments passed to ggplot2::scale_y_continuous() or ggplot2::scale_y_log10()

observedDataDirection
: Either "x" or "y", defining the direction of observed data.

lloqOnBothAxes
: A boolean; if TRUE, LLOQ lines are drawn on both axes. If FALSE (default), the LLOQ line is drawn for the observed-data axis only. (geom_vline when observedDataDirection = "x", geom_hline when observedDataDirection = "y").

yDisplayAsAbsolute
: A boolean that defines the direction of comparison lines.

Value

A ggplot object representing the plotted data.

See Also

Other plot functions: plotBoxWhisker(), plotForest(), plotHistogram(), plotPredVsObs(), plotQQ(), plotRangeDistribution(), plotRatioVsCov(), plotResVsCov(), plotTimeProfile()

print.ggWatermark *Print method for ggWatermark objects*

Description

This function customizes the printing of ggplot objects with the class "ggWatermark" by adding a watermark.

Usage

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

Arguments

x	A ggWatermark object created by ggplotWithWatermark().
...	Additional arguments to be passed to the print method, allowing for further customization of the output.

Value

A ggplot object with a watermark drawn on it. The watermark is displayed according to the specified options.

See Also

Other watermark: addWatermark(), ggplotWithWatermark(), plot.ggWatermark()

resetDefaultColorMapDistinct *reset the default color map for discrete colors*

Description

reset the default color map for discrete colors

Usage

```
resetDefaultColorMapDistinct(oldColorMaps)
```

Arguments

oldColorMaps	list of color maps previously set

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), getOspsuite.plots.option(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

---

resetDefaults *restore to previously stored settings*

Description

restore to previously stored settings

Usage

```
resetDefaults(oldDefaults)
```

Arguments

oldDefaults	listwith previously stored settings

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaultTheme(), setDefaultColorMapDist setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

---

resetDefaultTheme *reset the default theme*

Description

wrapper for ggplot2::theme_set(oldTheme)

Usage

```
resetDefaultTheme(oldTheme)
```

Arguments

oldTheme	theme to set

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaults(), setDefaultColorMapDistinct setDefaultTheme(), setDefaults(), setOspsuite.plots.option()

scale_shape_osp *OSP Shape Scale*

Description

Discrete shape scale that automatically assigns shapes from `ospShapeNames` in order based on the number of factor levels. Equivalent to `ggplot2::scale_shape()`.

If there are more levels than available shapes, shapes are recycled and a warning is issued.

Usage

```
scale_shape_osp(...)
```

Arguments

`...`	Passed to `ggplot2::discrete_scale`.

Value

A ggplot2 scale that can be added to a plot.

See Also

Other shapes: `Shapes`, `ospShapeNames`, `scale_shape_osp_identity()`, `scale_shape_osp_manual()`

Examples

```
library(ggplot2)
df <- data.frame(x = 1:3, y = 1:3, group = c("A", "B", "C"))
ggplot(df, aes(x, y, shape = group)) +
  geom_point_osp(size = 4) +
  scale_shape_osp()
```

scale_shape_osp_identity *OSP Shape Identity Scale*

Description

Identity scale for when data already contains OSP shape names. Use this when your shape column contains values from `ospShapeNames` directly (e.g., "circle", "diamond", "star"). Equivalent to `ggplot2::scale_shape_identity()`.

Usage

```
scale_shape_osp_identity(guide = "none", ...)
```

Arguments

guide	Guide for the legend. Use "legend" to show a legend, or "none" to hide it.
...	Passed to ggplot2::scale_shape_manual.

Value

A ggplot2 scale that can be added to a plot.

See Also

Other shapes: Shapes, ospShapeNames, scale_shape_osp(), scale_shape_osp_manual()

Examples

```
library(ggplot2)
df <- data.frame(x = 1:3, y = 1:3, shape = c("circle", "diamond", "star"))
ggplot(df, aes(x, y, shape = shape)) +
  geom_point_osp(size = 4) +
  scale_shape_osp_identity(guide = "legend")
```

---

scale_shape_osp_manual *OSP Shape Manual Scale*

Description

Manual shape scale for explicit mapping of factor levels to OSP shape names. Equivalent to ggplot2::scale_shape_manual().

Usage

```
scale_shape_osp_manual(values, ...)
```

Arguments

values	Named character vector. Names are factor levels; values are entries from ospShapeNames.
...	Passed to ggplot2::scale_shape_manual.

Value

A ggplot2 scale that can be added to a plot.

See Also

Other shapes: Shapes, ospShapeNames, scale_shape_osp(), scale_shape_osp_identity()

Examples

```
library(ggplot2)
df <- data.frame(x = 1:3, y = 1:3, group = c("A", "B", "C"))
ggplot(df, aes(x, y, shape = group)) +
  geom_point_osp(size = 4) +
  scale_shape_osp_manual(values = c(A = "circle", B = "diamond", C = "star"))
```

`setDefaultColorMapDistinct` *set the default color-map for discrete colors*

Description

Sets default color mappings for discrete color and fill aesthetics in ggplot2. Each color map should be a vector of valid color values (hex codes, color names, etc.).

Usage

```
setDefaultColorMapDistinct(colorMapList = NULL)
```

Arguments

`colorMapList`	list of color-maps to be set

Value

list with color-maps previously set

See Also

Other setDefault functions: `colorMaps`, `getDefaultGeomAttributes()`, `getDefaultOptions()`, `getOspsuite.plots.option()`, `resetDefaultColorMapDistinct()`, `resetDefaultTheme()`, `resetDefaults()`, `setDefaultTheme()`, `setDefaults()`, `setOspsuite.plots.option()`

Examples

```
## Not run:
# Set custom color maps
customColors <- list(
  c("#FF0000", "#00FF00", "#0000FF"), # RGB colors
  c("red", "green", "blue") # Named colors
)
oldColors <- setDefaultColorMapDistinct(customColors)

# Use default OSP color maps
setDefaultColorMapDistinct()

# Reset to previous colors
resetDefaultColorMapDistinct(oldColors)
```

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

`setDefaults` *sets the defaults for the OSPSuite.plots package*

Description

should be started at the beginning at each workflow

Usage

```
setDefaults(defaultOptions = list(), colorMapList = NULL)
```

Arguments

`defaultOptions`	list of options
`colorMapList`	list of color maps

Details

for detailed information see `vignette("ospsuite.plots", package = "ospsuite.plots")`

Value

list of old settings which can be used to reset defaults with `resetDefaults()`

See Also

Other setDefault functions: `colorMaps`, `getDefaultGeomAttributes()`, `getDefaultOptions()`, `getOspsuite.plots.option()`, `resetDefaultColorMapDistinct()`, `resetDefaultTheme()`, `resetDefaults()`, `setDefaultColorMapDistinct()`, `setDefaultTheme()`, `setOspsuite.plots.option()`

`setDefaultTheme` *set the default theme*

Description

set properties of the default theme for OSPSuite plots. This function applies a custom theme based on theme_bw() with OSPSuite-specific styling.

Usage

```
setDefaultTheme()
```

Value

invisibly return the previous theme so you can easily save it, then later restore it.

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaults(), setOspsuite.plots.option()

Examples

```
## Not run:
oldTheme <- setDefaultTheme()

# Create a plot with the new theme
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
print(p)

# Restore previous theme
resetDefaultTheme(oldTheme)

## End(Not run)
```

---

setOspsuite.plots.option

Set OSPSuite plots option with a given key and value.

Description

Set OSPSuite plots option with a given key and value.

Usage

```
setOspsuite.plots.option(optionKey, value)
```

Arguments

optionKey	The key for the option.
value	The value for the option.

See Also

Other setDefault functions: colorMaps, getDefaultGeomAttributes(), getDefaultOptions(), getOspsuite.plots.option(), resetDefaultColorMapDistinct(), resetDefaultTheme(), resetDefaults(), setDefaultColorMapDistinct(), setDefaultTheme(), setDefaults()

Examples

```
## Not run:
setOspsuite.plots.option(optionKey = OptionKeys$watermarkEnabled, value = TRUE)

## End(Not run)
```

`Shapes` *Shapes*

Description

Named list of OSP shape names for backward compatibility. Use `Shapes$circle` to get the shape name "circle".

Usage

```
Shapes
```

See Also

Other shapes: `ospShapeNames`, `scale_shape_osp()`, `scale_shape_osp_identity()`, `scale_shape_osp_manual()`

`stat_qq_osp` *OSP Q-Q Stat*

Description

A stat_qq that uses OSP shapes via `GeomPointOsp` instead of standard `geom_point`. This ensures QQ plots have visual consistency with other OSP plots.

Unlike `ggplot2::stat_qq()`, this function does not expose a `geom` parameter as it always uses `GeomPointOsp` for rendering.

Usage

```
stat_qq_osp(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  distribution = stats::qnorm,
  dparams = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
...	Other arguments passed on to layer()'s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can *not* be passed through .... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom's documentation lists which parameters it can accept. • Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat's documentation lists which parameters it can accept. • The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

distribution	Distribution function to use, if x not specified
dparams	Additional parameters passed on to distribution function.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. annotation_borders().

Value

A ggplot2 layer that can be added to a plot.

See Also

Other layers: GeomErrorbarOsp, GeomPointOsp, geom_errorbar_osp(), geom_point_osp()

---

updateScaleArgumentsForTimeUnit

adjust arguments for scale if dimension of scale is time

Description

adds break function with fixed width for breaks depending on unit:

Usage

```
updateScaleArgumentsForTimeUnit(scaleArgs, dimension, unit)
```

Arguments

scaleArgs	list of arguments for scale to be updated, passed to scale_x_continuous or scale_x_log10
dimension	dimension of axis, if not 'time' list will not be updated
unit	A named list of information about the data such as the dimension and unit of its variables.

Details

- s: width = 15,
- min: width = 15,
- h: width = 6,
- day(s): width = 7
- week(s): width = 4
- month(s): width = 6

The function uses the following logic to determine the breaks:

- If the range of time values is relatively small (i.e., less than twice the width of the breaks), it will use a default set of extended breaks.
- If the range of time values is larger, the function will check if it is appropriate to use wider breaks. Specifically, it will continue to double the width until it finds a width that is suitable, ensuring that 10 times the width is still less than the total range of time values. This means that the breaks will be spaced far enough apart to be meaningful without overcrowding the axis, providing clarity in the visualization.

Value

update list of arguments for scale

Examples

```
xScaleArgs <- list(limits = c(0, 24))
xScaleArgs <-
  updateScaleArgumentsForTimeUnit(
    scaleArgs = xScaleArgs,
    dimension = "time",
    unit = "h"
  )
addXScale(plotObject = ggplot(), xScale = "linear", xScaleArgs = xScaleArgs)
```

Index