

Package: tlf (via r-universe)

October 11, 2026

Type Package

Title TLF Library

Version 1.6.2

Description An implementation of the Table, Listing and Figure concepts in R. This library provides a standardized approach to creating tables and graphs based on conventions over configurations.

License GPL-2 | file LICENSE

Language en-US

URL <https://github.com/open-systems-pharmacology/tlf-library>

BugReports <https://github.com/open-systems-pharmacology/tlf-library/issues>

Imports ggplot2 (>= 3.3.0), R6, jsonlite, ospsuite.utils (>= 1.6.0), patchwork, ggtext, stringr (>= 1.5.0), rlang, lifecycle

Depends R (>= 4.1)

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

Encoding UTF-8

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Suggests cowplot, knitr, rmarkdown, scales, shiny, showtext, sysfonts, svglite, testthat (>= 3.0.3), vdiff (>= 1.0.0), devtools

VignetteBuilder knitr

Collate 'aaa-utilities.R' 'aggregation-input.R'
'aggregation-summary.R' 'atom-plots.R'
'boxwhisker-datamapping.R' 'boxwhisker-get-measure.R'
'boxwhisker-plotconfiguration.R'
'cumulative-time-profile-datamapping.R'
'datamapping-grouping.R' 'datamapping-groupmapping.R'
'datamapping-range.R' 'datamapping-xy.R'
'datamapping-xygroup.R' 'ddiratio-datamapping.R'
'error-checks.R' 'font.R' 'global-vars.R' 'helpers.R'

'histogram-datamapping.R' 'label.R' 'messages.R'
 'obs-vs-pred-datamapping.R' 'observed-data-mapping.R'
 'pkratio-datamapping.R' 'pkratio-get-measure.R'
 'plot-boxwhisker.R' 'plot-cumulative-time-profile.R'
 'plot-didiratio.R' 'utilities-enums.R' 'plot-grid.R'
 'plot-histogram.R' 'plot-obs-vs-pred.R'
 'plot-observed-time-profile.R' 'plot-piechart.R'
 'plot-pkratio.R' 'plot-qq.R' 'plot-res-vs-pred.R'
 'plot-simulated-time-profile.R' 'plot-timeprofile.R'
 'plot-tornado.R' 'plotconfiguration-axis.R'
 'plotconfiguration-background.R' 'plotconfiguration-export.R'
 'plotconfiguration-label.R' 'plotconfiguration-legend.R'
 'plotconfiguration-sub-classes.R' 'plotconfiguration.R'
 'qq-datamapping.R' 'themes.R' 'timeprofile-datamapping.R'
 'timeprofile-helper.R' 'timeprofile-plotconfiguration.R'
 'tlf-env.R' 'tornado-datamapping.R'
 'tornado-plotconfiguration.R' 'utilities-aesthetics.R'
 'utilities-axis.R' 'utilities-background.R'
 'utilities-export.R' 'utilities-ggplot.R' 'utilities-label.R'
 'utilities-legend.R' 'utilities-mapping.R'
 'utilities-molecule-plots.R' 'utilities-theme.R'
 'utilities-tlf.R' 'utils.R' 'zzz.R'

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libjpeg-dev libpng-dev libxml2-dev libssl-dev

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

Date/Publication 2025-11-19 10:22:02 UTC

RemoteUrl <https://github.com/Open-Systems-Pharmacology/TLF-Library>

RemoteRef v1.6.2

RemoteSha 68303a85f41fc26e8508e52a94dcb989b9a86cd2

Contents

.sanitizeLabel	7
addErrorbar	7
addLine	9
addRibbon	11
addScatter	13
addWatermark	15
AestheticFields	17
AestheticProperties	17
AestheticSelectionKeys	18
AggregationInput	18
AggregationSummary	19
Alignments	21
asLabel	22
AtomPlots	22

AxisConfiguration	23
BackgroundConfiguration	25
BackgroundElement	27
BoxWhiskerDataMapping	28
BoxWhiskerPlotConfiguration	30
ColorMaps	31
ColorPalettes	32
createWatermarkGrob	32
CumulativeTimeProfileDataMapping	33
CumulativeTimeProfilePlotConfiguration	34
DataMappings	35
DDIRatioDataMapping	35
DDIRatioPlotConfiguration	36
DefaultDataMappingValues	37
Directions	38
ExportConfiguration	38
ExportFormats	40
exportPlot	41
exportPlotConfigurationCode	42
ExportUnits	42
Font	43
FontFaces	45
geomTLFPoint	46
getBoxWhiskerMeasure	46
getDefaultCaptions	47
getDualAxisPlot	48
getGreekTickLabels	49
getGuestValues	49
getGuestValuesFromDataMapping	50
getLabelWithUnit	51
getLegendCaption	52
getLegendPosition	52
getLinesFromFoldDistance	53
getLnTickLabels	53
getLogTickLabels	54
getPercentileTickLabels	55
getPiTickLabels	55
getPKRatioMeasure	56
getSameLimits	57
getSqrtTickLabels	57
getSymmetricLimits	58
getTLFSettings	58
Grouping	59
GroupMapping	60
HistogramDataMapping	61
HistogramPlotConfiguration	62
HorizontalJustification	63
initializePlot	64

isBetween	65
Label	65
LabelConfiguration	68
LegendConfiguration	69
LegendPositions	71
LegendTypes	71
LineElement	72
Linetypes	73
loadThemeFromJson	74
loadTLFSettings	74
mean-1.96sd	75
mean-sd	75
mean+1.96sd	76
mean+sd	77
median-1.5IQR	78
median-IQR	78
median+1.5IQR	79
median+IQR	80
MoleculePlots	80
ObservedDataMapping	81
ObsVsPredDataMapping	83
ObsVsPredPlotConfiguration	85
Percentile0%	86
Percentile1%	87
Percentile10%	88
Percentile100%	89
Percentile15%	89
Percentile2.5%	90
Percentile20%	91
Percentile25%-1.5IQR	92
Percentile25%	92
Percentile5%	93
Percentile50%	94
Percentile75%+1.5IQR	95
Percentile75%	95
Percentile80%	96
Percentile85%	97
Percentile90%	98
Percentile95%	98
Percentile97.5%	99
Percentile99%	100
PieChartDataMapping	101
PieChartPlotConfiguration	101
PKRatioDataMapping	103
PKRatioPlotConfiguration	104
PlotAnnotationTextSize	105
plotBoxWhisker	105
PlotConfiguration	107

PlotConfigurations	110
plotCumulativeTimeProfile	111
plotDDIRatio	112
plotGrid	114
PlotGridConfiguration	115
plotHistogram	120
plotObservedTimeProfile	122
plotObsVsPred	123
plotPieChart	125
plotPKRatio	127
plotQQ	128
plotResVsPred	129
plotResVsTime	131
plotSimulatedTimeProfile	132
plotTimeProfile	133
plotTornado	134
QQDataMapping	136
QQPlotConfiguration	137
RangeDataMapping	138
resetTLFSettingsToDefault	139
ResVsPredDataMapping	140
ResVsPredPlotConfiguration	140
ResVsTimeDataMapping	141
ResVsTimePlotConfiguration	142
runPlotMaker	143
runThemeMaker	143
saveThemeToJson	144
saveTLFSettings	144
Scaling	144
setBackground	145
setBackgroundPanelArea	146
setBackgroundPlotArea	147
setCaptionColor	148
setCaptionFill	149
setCaptionLabels	149
setCaptionLinetype	150
setCaptionOrder	150
setCaptionShape	151
setCaptionSize	151
setCaptionVisibility	152
setDefaultAggregationBins	152
setDefaultAggregationFunctions	153
setDefaultAggregationLabels	153
setDefaultAlphaRatio	154
setDefaultErrorbarCapSize	154
setDefaultExportParameters	154
setDefaultLegendPosition	155
setDefaultLegendTitle	156

setDefaultLLOQLinetype	156
setDefaultLogTicks	156
setDefaultMaxCharacterWidth	157
setDefaultWatermark	157
setGrid	158
setLegend	158
setLegendCaption	159
setLegendFont	160
setLegendPosition	160
setLegendTitle	161
setPlotExport	161
setPlotExportDimensions	162
setPlotExportFormat	163
setPlotExportSize	163
setPlotLabels	164
setWatermark	165
setXAxis	166
setXGrid	167
setY2Axis	168
setYAxis	169
setYGrid	170
Shapes	171
TagPositions	172
Theme	172
ThemeAestheticMaps	173
ThemeAestheticSelections	175
ThemeBackground	176
ThemeFont	178
ThemePlotConfigurations	180
TickLabelTransforms	181
TimeProfileDataMapping	181
TimeProfilePlotConfiguration	184
tlfSettingsNames	185
tlfStatFunctions	186
TornadoDataMapping	186
TornadoPlotConfiguration	188
updateExportDimensionsForLegend	189
updateTimeProfileLegend	189
useDarkTheme	190
useExcelTheme	190
useHighChartTheme	191
useMatlabTheme	191
useMinimalTheme	191
useTemplateTheme	192
useTheme	192
VerticalJustification	192
XAxisConfiguration	193
XYDataMapping	194

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>x</code>	Numeric values to plot along the x axis. Only used instead of data if data is NULL.
<code>ymin</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.
<code>ymax</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.
<code>caption</code>	Optional character values defining the legend captions of the plot.
<code>color</code>	Optional character values defining the colors of the plot layer. See <code>grDevices::colors()</code> to get names of colors
<code>size</code>	Optional numeric values defining the size of the plot layer.
<code>linetype</code>	Optional character values defining the linetype of the plot layer. See <code>enum Linetypes</code> to get names of linetype.
<code>capSize</code>	Numeric extent of the error bars caps Caution the value corresponds to the ratio of the mean spacing between plotted error bars. For instance, an extent of 1 will fill the caps until the next error bar
<code>dataMapping</code>	A <code>RangeDataMapping</code> object mapping x, ymin, ymax and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional <code>PlotConfiguration</code> object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/atom-plots.html>

See Also

Other atom plots: `addLine()`, `addRibbon()`, `addScatter()`, `initializePlot()`

Examples

```
# Add errorbar using x, ymin and ymax
addErrorbar(
  x = c(1, 2, 1, 2, 3),
  ymin = c(5, 0, 2, 3, 4),
  ymax = c(6, 2, 6, 2.5, 5)
```

```
)

# Add errorbar using a data.frame
time <- seq(0, 30, 0.5)
errorbarData <- data.frame(x = time, ymin = cos(time) - 1, ymax = cos(time) + 1)

addErrorbar(
  data = errorbarData,
  dataMapping = RangeDataMapping$new(x = "x", ymin = "ymin", ymax = "ymax")
)

# Or for simple cases a smart mapping will get directly x, ymin and ymax from data
addErrorbar(data = errorbarData)

# Add a errorbar with caption
addErrorbar(data = errorbarData, caption = "My errorbar plot")

# Add a errorbar with specific properties
addErrorbar(data = errorbarData, color = "blue", size = 0.5, caption = "My data")

# Add a errorbar with specific properties
p <- addErrorbar(
  data = errorbarData,
  color = "blue", size = 0.5, caption = "My data"
)
addScatter(
  x = time, y = cos(time),
  color = "red", size = 1, caption = "My data",
  plotObject = p
)
```

addLine

addLine

Description

Add a line layer to a ggplot object.

Usage

```
addLine(
  data = NULL,
  metaData = NULL,
  x = NULL,
  y = NULL,
  caption = NULL,
  color = NULL,
  shape = NULL,
  size = NULL,
```

```

    linetype = NULL,
    dataMapping = NULL,
    plotConfiguration = NULL,
    plotObject = NULL
)

```

Arguments

<code>data</code>	A <code>data.frame</code> to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>x</code>	Numeric values to plot along the x axis. Only used instead of data if data is <code>NULL</code> .
<code>y</code>	Numeric values to plot along the y axis. Only used instead of data if data is <code>NULL</code> .
<code>caption</code>	Optional character values defining the legend captions of the plot.
<code>color</code>	Optional character values defining the colors of the plot layer. See <code>grDevices::colors()</code> to get names of colors
<code>shape</code>	Optional character values defining the shapes/symbols of the plot layer. See <code>enum Shapes</code> to get names of shapes.
<code>size</code>	Optional numeric values defining the size of the plot layer.
<code>linetype</code>	Optional character values defining the linetype of the plot layer. See <code>enum Linetypes</code> to get names of linetype.
<code>dataMapping</code>	A <code>XYGDataMapping</code> object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional <code>PlotConfiguration</code> object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional <code>ggplot</code> object on which to add the plot layer

Value

A `ggplot` object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/atom-plots.html>

See Also

Other atom plots: [addErrorbar\(\)](#), [addRibbon\(\)](#), [addScatter\(\)](#), [initializePlot\(\)](#)

Examples

```
# Add line using x and y
addLine(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

# Add line using a data.frame
time <- seq(0, 30, 0.1)
lineData <- data.frame(x = time, y = cos(time))

addLine(
  data = lineData,
  dataMapping = XYGDataMapping$new(x = "x", y = "y")
)

# Or for simple cases a smart mapping will get directly x and y from data
addLine(data = lineData)

# Add a line with caption
addLine(data = lineData, caption = "My line plot")

# Add a line with specific properties
addLine(
  data = lineData,
  color = "blue", linetype = "longdash", size = 0.5, caption = "My data"
)

# Add a line with specific properties
p <- addLine(
  data = lineData,
  color = "blue", linetype = "longdash", size = 0.5, caption = "My data"
)
addLine(
  x = c(0, 1), y = c(1, 0),
  color = "red", linetype = "solid", size = 1,
  plotObject = p
)
```

`addRibbon`*addRibbon*

Description

Add a ribbon layer to a ggplot object.

Usage

```
addRibbon(
  data = NULL,
  metaData = NULL,
```

```

x = NULL,
ymin = NULL,
ymax = NULL,
caption = NULL,
fill = NULL,
color = NULL,
size = NULL,
linetype = NULL,
alpha = NULL,
dataMapping = NULL,
plotConfiguration = NULL,
plotObject = NULL
)

```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>x</code>	Numeric values to plot along the x axis. Only used instead of data if data is NULL.
<code>ymin</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.
<code>ymax</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.
<code>caption</code>	Optional character values defining the legend captions of the plot.
<code>fill</code>	Optional character values defining the colors of the plot layer. See <code>grDevices::colors()</code> to get names of colors
<code>color</code>	Optional character values defining the colors of the plot layer. See <code>grDevices::colors()</code> to get names of colors
<code>size</code>	Optional numeric values defining the size of the plot layer.
<code>linetype</code>	Optional character values defining the linetype of the plot layer. See <code>enum Linetypes</code> to get names of linetype.
<code>alpha</code>	Numeric value between 0 and 1 corresponding to transparency of the ribbon. The closer to 0, the more transparent the ribbon is. The closer to 1, the more opaque the ribbon is.
<code>dataMapping</code>	A <code>RangeDataMapping</code> object mapping x, ymin, ymax and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional <code>PlotConfiguration</code> object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional <code>ggplot</code> object on which to add the plot layer

Value

A `ggplot` object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/atom-plots.html>

See Also

Other atom plots: `addErrorbar()`, `addLine()`, `addScatter()`, `initializePlot()`

Examples

```
# Add ribbon using x, ymin and ymax
addRibbon(
  x = c(1, 2, 1, 2, 3),
  ymin = c(5, 0, 2, 3, 4),
  ymax = c(6, 2, 6, 2.5, 5)
)

# Add ribbon using a data.frame
time <- seq(0, 30, 0.1)
ribbonData <- data.frame(x = time, ymin = cos(time) - 1, ymax = cos(time) + 1)

addRibbon(
  data = ribbonData,
  dataMapping = RangeDataMapping$new(x = "x", ymin = "ymin", ymax = "ymax")
)

# Or for simple cases a smart mapping will get directly x, ymin and ymax from data
addRibbon(data = ribbonData)

# Add a ribbon with caption
addRibbon(data = ribbonData, caption = "My ribbon plot")

# Add a ribbon with specific properties
addRibbon(data = ribbonData, fill = "blue", alpha = 0.5, caption = "My data")

# Add a ribbon with specific properties
p <- addRibbon(data = ribbonData, fill = "blue", alpha = 0.5, caption = "My data")
addRibbon(
  x = c(0, 1), ymin = c(-0.5, -0.5), ymax = c(0.5, 0.5),
  fill = "red", alpha = 1,
  plotObject = p
)
```

`addScatter`*addScatter*

Description

Add a scatter plot layer to a ggplot object

Usage

```
addScatter(
  data = NULL,
  metaData = NULL,
  x = NULL,
  y = NULL,
  caption = NULL,
  color = NULL,
  shape = NULL,
  size = NULL,
  linetype = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>x</code>	Numeric values to plot along the x axis. Only used instead of data if data is NULL.
<code>y</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.
<code>caption</code>	Optional character values defining the legend captions of the plot.
<code>color</code>	Optional character values defining the colors of the plot layer. See <code>grDevices::colors()</code> to get names of colors
<code>shape</code>	Optional character values defining the shapes/symbols of the plot layer. See <code>enum Shapes</code> to get names of shapes.
<code>size</code>	Optional numeric values defining the size of the plot layer.
<code>linetype</code>	Optional character values defining the linetype of the plot layer. See <code>enum Linetypes</code> to get names of linetype.
<code>dataMapping</code>	A <code>XYGDataMapping</code> object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional <code>PlotConfiguration</code> object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional <code>ggplot</code> object on which to add the plot layer

Value

A `ggplot` object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/atom-plots.html>

See Also

Other atom plots: `addErrorbar()`, `addLine()`, `addRibbon()`, `initializePlot()`

Examples

```
# Add scatter using x and y
addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

# Add scatter using a data.frame
time <- seq(0, 30, 0.1)
scatterData <- data.frame(x = time, y = cos(time))

addScatter(
  data = scatterData,
  dataMapping = XYGDataMapping$new(x = "x", y = "y")
)

# Or for simple cases a smart mapping will get directly x and y from data
addScatter(data = scatterData)

# Add a scatter with caption
addScatter(data = scatterData, caption = "My scatter plot")

# Add a scatter with specific properties
addScatter(
  data = scatterData,
  color = "blue", shape = "diamond", size = 2, caption = "My data"
)

# Add a scatter with specific properties
p <- addScatter(
  data = scatterData,
  color = "blue", shape = "diamond", size = 2, caption = "My data"
)
addScatter(
  x = c(0, 1), y = c(1, 0),
  color = "red", shape = "circle", size = 3,
  plotObject = p
)
```

Description

Creates a ggplot grob based on the label text and its font properties. Then, adds the grob to the ggplot object as a new layer using `ggplot2::annotation_custom`.

Usage

```
addWatermark(
  plotObject,
  watermark,
  color = NULL,
  size = NULL,
  angle = NULL,
  alpha = NULL
)
```

Arguments

<code>plotObject</code>	A ggplot object
<code>watermark</code>	A character value or a Label object
<code>color</code>	Color of the watermark.
<code>size</code>	Size of the watermark.
<code>angle</code>	Angle of the watermark (in degree).
<code>alpha</code>	Numeric value between 0 and 1 corresponding to transparency of the watermark The closer to 0, the more transparent the watermark is. The closer to 1, the more opaque the watermark is.

Value

A ggplot object

Examples

```
# Add a watermark to an empty plot
p <- ggplot2::ggplot()
addWatermark(p, "watermark")

# Watermark with font properties
watermarkLabel <- Label$new(text = "watermark", color = "blue")
addWatermark(p, watermarkLabel)

# Horizontal watermark
addWatermark(p, watermarkLabel, angle = 0)

# Watermark totally opaque
addWatermark(p, watermarkLabel, alpha = 1)

# As multiple layers of watermark:
p2 <- addWatermark(p, watermarkLabel, alpha = 1)
addWatermark(p2, "other watermark", color = "red", angle = 90)
```

AestheticFields	<i>AestheticFields</i>
-----------------	------------------------

Description

List of all available aesthetic fields that manage aesthetic properties

Usage

AestheticFields

Format

An object of class list of length 4.

See Also

Other enum helpers: [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

AestheticProperties	<i>AestheticProperties</i>
---------------------	----------------------------

Description

Enum of aesthetic property names of ggplot2

Usage

AestheticProperties

Format

An object of class list of length 6.

See Also

Other enum helpers: [AestheticFields](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

`AestheticSelectionKeys`*AestheticSelectionKeys*

Description

List of some ggplot2 shapes

Usage

`AestheticSelectionKeys`

Format

An object of class `list` of length 4.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

`AggregationInput`*AggregationInput*

Description

R6 class to be used to construct inputs to the `AggregationSummary` class

Public fields

`aggregationFunction` list of functions to use for aggregation

`aggregationFunctionName` vector of function names that will be used as variable name of the aggregation

`aggregationUnit` unit of aggregation output

`aggregationDimension` dimension of aggregation output

Methods

Public methods:

- `AggregationInput$new()`
- `AggregationInput$clone()`

Method `new()`: Create a new `AggregationInput` object

Usage:

```
AggregationInput$new(
  aggregationFunction = NULL,
  aggregationFunctionName = NULL,
  aggregationUnit = NULL,
  aggregationDimension = NULL
)
```

Arguments:

`aggregationFunction` list of functions to use for aggregation

`aggregationFunctionName` vector of function names that will be used as variable name of the aggregation

`aggregationUnit` unit of aggregation output

`aggregationDimension` dimension of aggregation output

Returns: A new `AggregationInput` object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

AggregationSummary	<i>AggregationSummary</i>
--------------------	---------------------------

Description

R6 class to split a `data.frame` data into subsets defined by unique combinations of elements in the columns `xColumnNames` and `groupingColumnNames`. Applies functions defined in `aggregationFunctionsVector` to column `yColumnNames`. Returns a list of `data.frame`, one `data.frame` for each function listed in `aggregationFunctionsVector`. Each `data.frame` in list element is named after the function's corresponding string in `aggregationFunctionNames`. The summary statistic column name in each `data.frame` is the same as the name of the `data.frame` in the returned list.

Public fields

data data.frame
 metaData list of information on data
 xColumnNames character names of grouping variables
 groupingColumnNames character names of grouping variables
 yColumnNames character names of dependent variables (that are grouped)
 aggregationInputsVector list of R6 class AggregationInput objects
 aggregationFunctionsVector list of functions to use for aggregation
 aggregationFunctionNames vector of function names that will be used as variable name of the aggregation
 aggregationUnitsVector character vector of units of aggregation output
 aggregationDimensionsVector character vector of dimensions of aggregation output
 dfHelper data.frame of aggregated values
 metaDataHelper list of information on dfHelper

Methods**Public methods:**

- [AggregationSummary\\$new\(\)](#)
- [AggregationSummary\\$applyAggregationFunctions\(\)](#)
- [AggregationSummary\\$generateAggregatedValues\(\)](#)
- [AggregationSummary\\$clone\(\)](#)

Method `new()`: Create a new AggregationSummary object

Usage:

```

AggregationSummary$new(
  data,
  metaData = NULL,
  xColumnNames = NULL,
  groupingColumnNames = NULL,
  yColumnNames = NULL,
  aggregationInputsVector = NULL,
  aggregationFunctionsVector = NULL,
  aggregationFunctionNames = NULL,
  aggregationUnitsVector = NULL,
  aggregationDimensionsVector = NULL
)

```

Arguments:

data data.frame
 metaData list of information on data
 xColumnNames character names of grouping variables
 groupingColumnNames character names of grouping variables

yColumnNames character names of dependent variables (that are grouped)
 aggregationInputsVector list of R6 class AggregationInput objects
 aggregationFunctionsVector list of functions to use for aggregation
 aggregationFunctionNames vector of function names that will be used as variable name of the aggregation
 aggregationUnitsVector character vector of units of aggregation output
 aggregationDimensionsVector character vector of dimensions of aggregation output
Returns: A new AggregationSummary object

Method applyAggregationFunctions(): Apply aggregation functions on x

Usage:

AggregationSummary\$applyAggregationFunctions(x)

Arguments:

x numeric vector

Returns: A list or vector of aggregated values

Method generateAggregatedValues(): Generate aggregated values

Usage:

AggregationSummary\$generateAggregatedValues()

Returns: A list or vector of aggregated values

Method clone(): The objects of this class are cloneable with this method.

Usage:

AggregationSummary\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Alignments

Alignments

Description

List of all available alignments/justifications for fonts

Usage

Alignments

Format

An object of class list of length 3.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

asLabel	<i>asLabel</i>
---------	----------------

Description

Set a character string into a Label object associating font properties to input If text is already a Label object, asLabel can be used to update its font properties

Usage

```
asLabel(text = "", font = NULL)
```

Arguments

text	A character value
font	A Font object defining the font properties of the Label

Value

A Label object

Examples

```
title <- "Title of Plot"
title <- asLabel(title)
```

AtomPlots	<i>AtomPlots</i>
-----------	------------------

Description

List of all available atom plots

Usage

```
AtomPlots
```

Format

An object of class list of length 5.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

AxisConfiguration

AxisConfiguration

Description

R6 class defining the configuration of axis

Active bindings

`valuesLimits` numeric vector of length 2 defining limits of axis. A value of NULL is allowed and lead to default ggplot2 behaviour

`axisLimits` numeric vector of length 2 defining limits of axis. A value of NULL is allowed and lead to default ggplot2 behaviour

`scale` name of axis scale from Enum Scaling A value of NULL is allowed and will lead to a default linear scale

`ticks` function or values defining where axis ticks are placed

`minorTicks` function or values defining where axis minor ticks are placed

`ticklabels` function or values defining the axis tick labels

`font` Font object defining the font of the ticklabels

`expand` logical defining if data is expanded until axis. If TRUE, data is expanded until axis If FALSE, some space between data and axis is kept

Methods**Public methods:**

- [AxisConfiguration\\$new\(\)](#)
- [AxisConfiguration\\$ggplotScale\(\)](#)
- [AxisConfiguration\\$ggplotExpansion\(\)](#)
- [AxisConfiguration\\$prettyTicks\(\)](#)
- [AxisConfiguration\\$prettyMinorTicks\(\)](#)
- [AxisConfiguration\\$prettyTickLabels\(\)](#)
- [AxisConfiguration\\$clone\(\)](#)

Method `new()`: Create a new AxisConfiguration object

Usage:

```
AxisConfiguration$new(
  valuesLimits = NULL,
  axisLimits = NULL,
  limits = lifecycle::deprecated(),
  scale = Scaling$lin,
  ticks = NULL,
  ticklabels = NULL,
  minorTicks = NULL,
  font = NULL,
  expand = FALSE
)
```

Arguments:

`valuesLimits` numeric vector of value limits (data outside these limits is removed)
`axisLimits` numeric vector of axis limits (data outside these limits is kept but not plotted)
`limits` **[Deprecated]**. Replaced by `axisLimits` argument.
`scale` character defining axis scale Use enum `Scaling` to access predefined scales.
`ticks` numeric vector or function defining where to position axis ticks
`ticklabels` character vector or function defining what to print on axis ticks
`minorTicks` numeric vector or function defining where to position minor axis ticks
`font` Font object defining the font of ticklabels
`expand` logical defining if data is expanded until axis. If TRUE, data is expanded until axis If FALSE, some space between data and axis is kept

Returns: A new AxisConfiguration object

Method `ggplotScale()`: Get the ggplot2 actual trans name of scale

Usage:

```
AxisConfiguration$ggplotScale()
```

Returns: A character included in ggplot2 available trans names

Method `ggplotExpansion()`: Get the ggplot2 actual function for expansion

Usage:

```
AxisConfiguration$ggplotExpansion()
```

Returns: A ggplot2 function

Method `prettyTicks()`: Get tick values for pretty default log plots

Usage:

```
AxisConfiguration$prettyTicks()
```

Returns: User defined tick values or tlf default ticks

Method `prettyMinorTicks()`: Get tick values for pretty default log plots

Usage:

```
AxisConfiguration$prettyMinorTicks()
```

Returns: User defined tick values or tlf default ticks

Method prettyTickLabels(): Get tick labels for pretty default log plots

Usage:

AxisConfiguration\$prettyTickLabels()

Returns: User defined tick labels or tlf default ticklabels

Method clone(): The objects of this class are cloneable with this method.

Usage:

AxisConfiguration\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

BackgroundConfiguration

BackgroundConfiguration

Description

R6 class defining the configuration of background

Active bindings

watermark Label object

plot BackgroundElement object

panel BackgroundElement object

xAxis LineElement object

yAxis LineElement object

y2Axis LineElement object

xGrid LineElement object

yGrid LineElement object

y2Grid LineElement object

Methods

Public methods:

- [BackgroundConfiguration\\$new\(\)](#)
- [BackgroundConfiguration\\$updatePlot\(\)](#)
- [BackgroundConfiguration\\$clone\(\)](#)

Method `new()`: Create a new BackgroundConfiguration object

Usage:

```
BackgroundConfiguration$new(
  watermark = NULL,
  plot = NULL,
  panel = NULL,
  xAxis = NULL,
  yAxis = NULL,
  y2Axis = NULL,
  xGrid = NULL,
  yGrid = NULL,
  y2Grid = NULL
)
```

Arguments:

`watermark` Label object defining properties of watermark

`plot` BackgroundElement object defining outside plot background properties

`panel` BackgroundElement object defining panel (inside of plot) background properties

`xAxis` LineElement object defining properties of x-axis

`yAxis` LineElement object defining properties of y-axis

`y2Axis` LineElement object defining properties of right y-axis

`xGrid` LineElement object defining properties of x-grid

`yGrid` LineElement object defining properties of y-grid

`y2Grid` LineElement object defining properties of right y-grid

Returns: A new BackgroundConfiguration object

Method `updatePlot()`: Update background a ggplot object from BackgroundConfiguration properties

Usage:

```
BackgroundConfiguration$updatePlot(plotObject)
```

Arguments:

`plotObject` a ggplot object

Returns: A ggplot object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

BackgroundElement

*BackgroundElement***Description**

R6 class defining the properties of background elements

Public fields

`fill` character defining the color filling of the background element

`color` character defining the color of the background element frame/line

`size` numeric defining the size of the background element frame/line

`linetype` character defining the size of the background element frame/line

Methods**Public methods:**

- [BackgroundElement\\$new\(\)](#)
- [BackgroundElement\\$createPlotElement\(\)](#)
- [BackgroundElement\\$clone\(\)](#)

Method `new()`: Create a new BackgroundElement object

Usage:

```
BackgroundElement$new(fill = NULL, color = NULL, size = NULL, linetype = NULL)
```

Arguments:

`fill` character color filling of the background element

`color` character color of the frame of the background element

`size` character size of the frame of the background element

`linetype` character linetype of the frame of the background element

Returns: A new BackgroundElement object

Method `createPlotElement()`: Create a `ggplot2::element_rect` directly usable by `ggplot2::theme`.

Usage:

```
BackgroundElement$createPlotElement(
  fill = NULL,
  color = NULL,
  size = NULL,
  linetype = NULL
)
```

Arguments:

fill character color filling of the background element

color character color of the frame of the background element

size character size of the frame of the background element

linetype character linetype of the frame of the background element

Returns: An element_rect object.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

BoxWhiskerDataMapping *BoxWhiskerDataMapping*

Description

R6 class for mapping y, GroupMapping, boxWhiskerLimits and outlierLimits to data

Super classes

```
tlf::XYDataMapping -> tlf::XYGDataMapping -> BoxWhiskerDataMapping
```

Public fields

outlierLimits List of minOutlierLimit and maxOutlierLimit functions outside which data is flagged as outlier

boxWhiskerLimits List of ymin, lower, middle, upper and ymax functions calculated on data to obtain box whiskers

Methods

Public methods:

- `BoxWhiskerDataMapping$new()`
- `BoxWhiskerDataMapping$getBoxWhiskerLimits()`
- `BoxWhiskerDataMapping$getOutliers()`
- `BoxWhiskerDataMapping$clone()`

Method `new()`: Create a new `BoxWhiskerDataMapping` object

Usage:

```
BoxWhiskerDataMapping$new(
  x = NULL,
  y,
  ymin = tlfStatFunctions$`Percentile5%`,
  lower = tlfStatFunctions$`Percentile25%`,
  middle = tlfStatFunctions$`Percentile50%`,
  upper = tlfStatFunctions$`Percentile75%`,
  ymax = tlfStatFunctions$`Percentile95%`,
  minOutlierLimit = tlfStatFunctions$`Percentile25%-1.5IQR`,
  maxOutlierLimit = tlfStatFunctions$`Percentile75%+1.5IQR`,
  ...
)
```

Arguments:

`x` Name of x variable to map Default value is NULL in case of a unique box in the boxplot.

`y` Name of y variable to map

`ymin` Name of function used for calculating lower whisker. Default value is `Percentile5%`.

`lower` Name of function used for calculating lower line of box Default value is `Percentile25%`.

`middle` Name of function used for calculating middle line Default value is `Percentile55%`.

`upper` Name of function used for calculating upper line of box Default value is `Percentile75%`.

`ymax` Name of function used for calculating upper whisker Default value is `Percentile95%`.

`minOutlierLimit` Name of function used for calculating lower outlier limit Default value is `Percentile25-1.5IQR%`.

`maxOutlierLimit` Name of function used for calculating upper outlier limit Default value is `Percentile75+1.5IQR%`.

`...` parameters inherited from `XYGDataMapping`

Returns: A new `BoxWhiskerDataMapping` object

Method `getBoxWhiskerLimits()`: Get a data.frame with box-whisker limit by group

Usage:

```
BoxWhiskerDataMapping$getBoxWhiskerLimits(data)
```

Arguments:

`data` data.frame to check

Returns: A data.frame with `ymin`, `lower`, `middle`, `upper`, `ymax` variables.

Method `getOutliers()`: Get a data.frame flagging outliers

Usage:

```
BoxWhiskerDataMapping$getOutliers(data)
```

Arguments:

data data.frame to check

Returns: A data.frame with minOutliers and maxOutliers variables. Values not flagged are NA in the outliers variables

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

BoxWhiskerPlotConfiguration

BoxWhiskerPlotConfiguration

Description

R6 class defining the configuration of a ggplot object for boxplots

Super class

```
tlf::PlotConfiguration -> BoxWhiskerPlotConfiguration
```

Public fields

defaultXScale Default xAxis scale value when creating a BoxWhiskerPlotConfiguration object

Active bindings

outliers logical defining if outliers should be included in boxplot

Methods**Public methods:**

- [BoxWhiskerPlotConfiguration\\$new\(\)](#)
- [BoxWhiskerPlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new `BoxWhiskerPlotConfiguration` object

Usage:

```
BoxWhiskerPlotConfiguration$new(outliers = TRUE, ...)
```

Arguments:

`outliers` logical defining if outliers should be included in boxplot

`...` parameters inherited from `PlotConfiguration`

Returns: A new `BoxWhiskerPlotConfiguration` object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `PlotConfiguration` classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

ColorMaps

ColorMaps

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
```

Format

An object of class `list` of length 6.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

ColorPalettes

ColorPalettes

Description

Enum of available color palettes First color palettes come from [viridis](#) Remaining color palettes are from [RColorBrewer](#)

Usage

ColorPalettes

Format

An object of class list of length 42.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

createWatermarkGrob

createWatermarkGrob

Description

Creates a ggplot grob based on the label text and its font properties.

Usage

createWatermarkGrob(label, alpha = NULL)

Arguments

label	A character value or Label object
alpha	Numeric value between 0 and 1 corresponding to transparency of the watermark The closer to 0, the more transparent the watermark is. The closer to 1, the more opaque the watermark is.

Value

A ggplot grob

CumulativeTimeProfileDataMapping

CumulativeTimeProfileDataMapping

Description

R6 class for mapping x, y, GroupMapping variables to data

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> CumulativeTimeProfileDataMapping`

Methods**Public methods:**

- `CumulativeTimeProfileDataMapping$clone()`

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

Usage:

`CumulativeTimeProfileDataMapping$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

CumulativeTimeProfilePlotConfiguration
CumulativeTimeProfilePlotConfiguration

Description

R6 class defining the configuration of a ggplot object for cumulative time profile plots

Super class

`tlf::PlotConfiguration -> CumulativeTimeProfilePlotConfiguration`

Public fields

`defaultExpand` Default expand value when creating a CumulativeTimeProfilePlotConfiguration object
`colorPalette` color palette property from ggplot2

Methods

Public methods:

- [CumulativeTimeProfilePlotConfiguration\\$new\(\)](#)
- [CumulativeTimeProfilePlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new CumulativeTimeProfilePlotConfiguration object

Usage:

`CumulativeTimeProfilePlotConfiguration$new(colorPalette = NULL, ...)`

Arguments:

`colorPalette` color palette property from ggplot2
`...` parameters inherited from PlotConfiguration

Returns: A new CumulativeTimeProfilePlotConfiguration object

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

Usage:

`CumulativeTimeProfilePlotConfiguration$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

DataMappings

DataMappings

Description

List of all available molecule plots

Usage

DataMappings

Format

An object of class list of length 18.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

DDIRatioDataMapping

DDIRatioDataMapping

Description

R6 class for mapping x, y, GroupMapping and DDI ratio lines variables to data

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> tlf::PKRatioDataMapping -> DDIRatioDataMapping`

Active bindings

`deltaGuest` Value of delta in [Guest et al.](#) equation

`minRange` Minimum range of x values for guest and ratio lines

`residualsVsObserved` Logical defining if calculated DDI data are as residuals vs observed or predicted vs observed

Methods

Public methods:

- [DDIRatioDataMapping\\$new\(\)](#)
- [DDIRatioDataMapping\\$clone\(\)](#)

Method `new()`: Create a new `DDIRatioDataMapping` object

Usage:

```
DDIRatioDataMapping$new(
  deltaGuest = NULL,
  minRange = c(0.01, 100),
  lines = DefaultDataMappingValues$ddiRatio,
  residualsVsObserved = FALSE,
  ...
)
```

Arguments:

`deltaGuest` Value of delta in [Guest et al.](#) equation. Default value is 1.

`minRange` Minimum range of x values for guest and ratio lines Default is [0.01 - 100]

`lines` List of ratio limits to display as diagonal/horizontal lines

`residualsVsObserved` Logical defining if calculated DDI data are as residuals vs observed or predicted vs observed

... parameters inherited from `PKRatioDataMapping`

Returns: A new `DDIRatioDataMapping` object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `DataMapping` classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

DDIRatioPlotConfiguration

DDIRatioPlotConfiguration

Description

R6 class defining the configuration of a `ggplot` object for DDI ratio plots

Super class

`tlf::PlotConfiguration -> DDIRatioPlotConfiguration`

Public fields

`defaultXScale` Default xAxis scale value when creating a DDIRatioPlotConfiguration object

`defaultYScale` Default yAxis scale value when creating a DDIRatioPlotConfiguration object

`defaultExpand` Default expand value when creating a DDIRatioPlotConfiguration object

Methods**Public methods:**

- [DDIRatioPlotConfiguration\\$clone\(\)](#)

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

Usage:

`DDIRatioPlotConfiguration$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

DefaultDataMappingValues

DefaultDataMappingValues

Description

List of default values used in dataMapping

Usage

DefaultDataMappingValues

Format

An object of class list of length 7.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Directions

Directions

Description

Enum of plotting directions for errorbars and lloq lines.

Usage

Directions

Format

An object of class list of length 3.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

ExportConfiguration

ExportConfiguration

Description

R6 class defining properties for saving a ggplot object

Public fields

name character defining the name of the file to be saved (without extension)
path Path of the directory to save plot to: path and filename are combined to create the fully qualified file name. Defaults to the working directory.
format character defining the format of the file to be saved
width numeric values defining the width in units of the plot dimensions after saving
height numeric values defining the height in units of the plot dimensions after saving
units character defining the unit of the saving dimension
dpi (dots per inch) numeric value defining plot resolution

Methods

Public methods:

- `ExportConfiguration$new()`
- `ExportConfiguration$print()`
- `ExportConfiguration$getFileName()`
- `ExportConfiguration$savePlot()`
- `ExportConfiguration$convertPixels()`
- `ExportConfiguration$clone()`

Method `new()`: Create a new `ExportConfiguration` object

Usage:

```
ExportConfiguration$new(  
  path = NULL,  
  name = NULL,  
  format = NULL,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL  
)
```

Arguments:

`path` Path of the directory to save plot to: path and filename are combined to create the fully qualified file name. Defaults to the working directory.

`name` character defining the name of the file to be saved (without extension)

`format` character defining the format of the file to be saved.

`width` numeric values defining the width in units of the plot dimensions after saving

`height` numeric values defining the height in units of the plot dimensions after saving

`units` character defining the unit of the saving dimension

`dpi` numeric value defining plot resolution (dots per inch)

Returns: A new `ExportConfiguration` object

Method `print()`: Print properties of export configuration

Usage:

```
ExportConfiguration$print()
```

Returns: Export configuration properties

Method `getFileName()`: Print the default exported file name from the export configuration

Usage:

```
ExportConfiguration$getFileName()
```

Returns: Default file name

Method `savePlot()`: Save/Export a plot

Usage:

```
ExportConfiguration$savePlot(plotObject, fileName = NULL)
```

Arguments:

plotObject A ggplot object

fileName character file name of the exported plot

Returns: The file name of the exported plot

Method convertPixels(): If unit is in pixels, convert all export dimensions to inches to keep compatibility with older versions of ggplot2

Usage:

```
ExportConfiguration$convertPixels()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

ExportFormats

ExportFormats

Description

List of all available formats to export a ggplot object

Usage

```
ExportFormats
```

Format

An object of class list of length 10.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

exportPlot	<i>exportPlot</i>
------------	-------------------

Description

Save a ggplot object according to its export properties

Usage

```
exportPlot(  
  plotObject,  
  fileName = NULL,  
  name = NULL,  
  format = NULL,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL  
)
```

Arguments

plotObject	Graphical object created from ggplot
fileName	name of exported file (with extension)
name	character defining the name of the file to be saved (without extension)
format	character defining the format of the file to be saved.
width	numeric values defining the width in units of the plot dimensions after saving
height	numeric values defining the height in units of the plot dimensions after saving
units	character defining the unit of the saving dimension
dpi	numeric value defining plot resolution (dots per inch)

Value

The file name of the exported plot

```
exportPlotConfigurationCode
      exportPlotConfigurationCode
```

Description

Export a plot configuration as R code

Usage

```
exportPlotConfigurationCode(plotConfiguration, name = "plotConfiguration")
```

Arguments

```
plotConfiguration
      A PlotConfiguration object
name
      Name of PlotConfiguration object in the created code
```

Value

R code to recreate the plot configuration as character

```
ExportUnits      ExportUnits
```

Description

List of all available units for width and height to export a ggplot object

Usage

```
ExportUnits
```

Format

An object of class list of length 4.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Font

*Font***Description**

R6 class defining font properties

Public fields

`size` numeric defining the size of font

`color` character defining the color of font

`fontFamily` character defining the family of font

`fontFace` character defining the font face as defined in helper enum `FontFaces`.

`angle` numeric defining the angle of font

`align` character defining the alignment of font as defined in helper enum `Alignments`.

`maxWidth` numeric that will be converted to a `ggplot2::unit` object (in "pt" unit) defining the maximum width of text box.

`margin` a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Methods**Public methods:**

- [Font\\$new\(\)](#)
- [Font\\$createPlotTextFont\(\)](#)
- [Font\\$createPlotTextBoxFont\(\)](#)
- [Font\\$clone\(\)](#)

Method `new()`: Create a new `Font` object. Default font properties are defined directly in the object field, so NULL input is allowed will lead to default properties.

Usage:

```
Font$new(
  color = NULL,
  size = NULL,
  fontFamily = NULL,
  fontFace = NULL,
  angle = NULL,
  align = NULL,
  maxWidth = NULL,
  margin = NULL
)
```

Arguments:

`color` character defining the color of font.

size numeric defining the size of font.
fontFamily character defining the family of font.
fontFace character defining the font face as defined in helper enum `FontFaces`.
angle numeric defining the angle of font.
align character defining the alignment of font as defined in helper enum `Alignments`.
maxWidth numeric that will be converted to a `ggplot2::unit` object (in "pt" unit) defining the maximum width of text box.
margin a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: A new `Font` object

Method `createPlotTextFont()`: Create a `ggplot2::element_text` directly convertible by `ggplot2::theme`.

Usage:

```
Font$createPlotTextFont(
  size = NULL,
  color = NULL,
  fontFamily = NULL,
  fontFace = NULL,
  angle = NULL,
  align = NULL,
  margin = NULL
)
```

Arguments:

size numeric defining the size of font
color character defining the color of font
fontFamily character defining the family of font
fontFace character defining the font face as defined in helper enum `FontFaces`.
angle numeric defining the angle of font.
align character defining the alignment of font as defined in helper enum `Alignments`.
margin a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: An `element_text` object.

Method `createPlotTextBoxFont()`: Create a `ggplot2::element_text` directly convertible by `ggplot2::theme`.

Usage:

```
Font$createPlotTextBoxFont(
  size = NULL,
  color = NULL,
  fontFamily = NULL,
  fontFace = NULL,
  angle = NULL,
  align = NULL,
  maxWidth = NULL,
  margin = NULL
)
```

Arguments:

size numeric defining the size of font
color character defining the color of font
fontFamily character defining the family of font
fontFace character defining the font face as defined in helper enum FontFaces.
angle numeric defining the angle of font.
align character defining the alignment of font as defined in helper enum Alignments.
maxWidth numeric that will be converted to a ggplot2::unit object (in "pt" unit) defining the maximum width of text box.
margin a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: An ggtext::element_textbox object.

Method clone(): The objects of this class are cloneable with this method.

Usage:

Font\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

FontFaces

FontFaces

Description

List of all available font faces

Usage

FontFaces

Format

An object of class list of length 4.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

geomTLFPoint	<i>geomTLFPoint</i>
--------------	---------------------

Description

geom similar to geom_point() but that leverage fonts to draw its shapes

Usage

```
geomTLFPoint(  
  mapping = NULL,  
  data = NULL,  
  stat = "identity",  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  ...  
)
```

Arguments

mapping	mapping from ggplot2 package as provided by aes()
data	data.frame
stat	stat name from ggplot2
position	position name from ggplot2
na.rm	a logical value indicating
show.legend	= NA,
inherit.aes	a logical value indicating if aesthetics are inherited
...	other arguments.

getBoxWhiskerMeasure	<i>getBoxWhiskerMeasure</i>
----------------------	-----------------------------

Description

Get a summary table of Box Whisker percentiles

Usage

```
getBoxWhiskerMeasure(
  data,
  dataMapping = NULL,
  y = NULL,
  group = NULL,
  quantiles = c(0.05, 0.25, 0.5, 0.75, 0.95)
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>dataMapping</code>	A BoxWhiskerDataMapping object mapping x, y and aesthetic groups to their variable names of data.
<code>y</code>	Name of y variable in data.
<code>group</code>	Name of grouping variable in data.
<code>quantiles</code>	Numeric values between 0 and 1 defining the quantiles to summarize

Value

A data.frame of summary statistics

Examples

```
# Get box-and-whisker plots of log-normal distributed data
boxData <- data.frame(x = c(rep("A", 500), rep("B", 500)), y = rlnorm(1000))

getBoxWhiskerMeasure(data = boxData, dataMapping = BoxWhiskerDataMapping$new(x = "x", y = "y"))
```

getDefaultCaptions	<i>getDefaultCaptions</i>
--------------------	---------------------------

Description

Creates default legend captions by concatenating the values of the data and metaData of variableList variables from data.

Usage

```
getDefaultCaptions(
  data,
  metaData = NULL,
  variableList = colnames(data),
  sep = "-"
)
```

Arguments

data	data.frame used for legend caption
metaData	list of lists containing metaData on data
variableList	ordered vector of variables used for specifying the caption
sep	characters separating variables in caption

Value

Factor levels corresponding to the legend captions

Examples

```
data <- data.frame(
  Population = c("Caucasian", "Asian", "Caucasian", "Asian"),
  Gender = c("Male", "Male", "Female", "Female"),
  Dose = c(50, 100, 100, 50),
  Compound = c("Midazolam", "Midazolam", "Midazolam", "Midazolam")
)

metaData <- list(Dose = list(unit = "mg"))

# Get captions using every variable of data
getDefaultCaptions(data, metaData)

# Get captions using specific variables of data
getDefaultCaptions(data, metaData, variableList = c("Gender", "Population"))

# Get captions separating variables witha space (character " ")
getDefaultCaptions(data, metaData, sep = " ")
```

getDualAxisPlot

getDualAxisPlot

Description

Check if dual Y Axis is needed

Usage

```
getDualAxisPlot(leftPlotObject, rightPlotObject)
```

Arguments

leftPlotObject	A ggplot object with left y-axis
rightPlotObject	A ggplot object with right y-axis

Value

A ggplot object with dual y-axis

getGreekTickLabels	<i>getGreekTickLabels</i>
--------------------	---------------------------

Description

Get ticklabels expressions for discrete scale plots with greek letters

Usage

```
getGreekTickLabels(ticks)
```

Arguments

ticks numeric values of the ticks

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- c(1, 5, 10, 50, 100, 500)
getGreekTickLabels(ticks)
```

getGuestValues	<i>getGuestValues</i>
----------------	-----------------------

Description

Get a data.frame with Guest et al. ratio limits with:

- $y_{max} = x \cdot limit$
- $y_{min} = x / limit$
- $limit = (\delta + 2(x-1))/x$

Usage

```
getGuestValues(x, delta = 1, residualsVsObserved = FALSE)
```

Arguments

<code>x</code>	Numeric values input of Guest function
<code>delta</code>	Numeric value parameter of Guest function
<code>residualsVsObserved</code>	Logical value defining if limits are calculated as residuals vs observed, instead of predicted vs observed.

Value

A data.frame with `x`, `ymin` and `ymax` defining Guest et al. limits

References

<https://dmd.aspetjournals.org/content/39/2/170>

Examples

```
# Get predicted vs observed Guest et al. limits
getGuestValues(x = 10^seq(-2, 2, 0.2))

# Get residuals vs observed Guest et al. limits
getGuestValues(x = 10^seq(-2, 2, 0.2), residualsVsObserved = TRUE)
```

```
getGuestValuesFromDataMapping
      getGuestValuesFromDataMapping
```

Description

Get a data.frame with Guest et al. ratio limits from data and its DDIRatioDataMapping

Usage

```
getGuestValuesFromDataMapping(data, dataMapping)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>dataMapping</code>	A DDIRatioDataMapping object mapping <code>x</code> , <code>y</code> and aesthetic groups to their variable names of data.

Value

A data.frame with `x`, `ymin` and `ymax` defining Guest et al. limits

Examples

```
# Get the data.frame of Guest et al. limits
ddiData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

getGuestValuesFromDataMapping(
  data = ddiData,
  dataMapping = DDIRatioDataMapping$new(x = "x", y = "y")
)
```

getLabelWithUnit	<i>getLabelWithUnit</i>
------------------	-------------------------

Description

Get label with its unit within square brackets when available

Usage

```
getLabelWithUnit(label, unit = NULL)
```

Arguments

- label text of axis label
- unit Character value corresponding to unit of label

Value

label [unit] or label depending if unit is NULL or ""

Examples

```
getLabelWithUnit("Time", "min")

getLabelWithUnit("Label without unit")
```

<code>getLegendCaption</code>	<i>getLegendCaption</i>
-------------------------------	-------------------------

Description

Get the legend caption

Usage

```
getLegendCaption(plotObject)
```

Arguments

`plotObject` ggplot graphical object

Value

A data.frame corresponding to legend caption

<code>getLegendPosition</code>	<i>getLegendPosition</i>
--------------------------------	--------------------------

Description

Get the legend position

Usage

```
getLegendPosition(plotObject)
```

Arguments

`plotObject` ggplot graphical object

Value

Position of legend as defined in enum LegendPositions

See Also

LegendPositions

getLinesFromFoldDistance	
	<i>getLinesFromFoldDistance</i>

Description

Get list of values to provide to lines argument of dataMapping objects. The lines are internally used as argument of geom_hline and geom_abline from ggplot2

Usage

```
getLinesFromFoldDistance(foldDistance)
```

Arguments

foldDistance	Numeric values Caution: this argument is meant for log scaled plots and since fold distance is a ratio it is expected positive. In particular, line of identity corresponds to a foldDistance of 1.
--------------	--

Value

A list of numeric values

Examples

```
# Get lines for identity and 2-fold distance
getLinesFromFoldDistance(c(1, 2))

# Create dataMapping with lines identity and 2-fold distance
dataMapping <- ObsVsPredDataMapping$new(
  x = "predicted",
  y = "observed",
  lines = getLinesFromFoldDistance(c(1, 2))
)
```

getLnTickLabels	<i>getLnTickLabels</i>
-----------------	------------------------

Description

Get ticklabels expressions for ln scale plots

Usage

```
getLnTickLabels(ticks)
```

Arguments

ticks numeric values of the ticks

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- exp(c(1, 5, 10, 50, 100, 500))
getLnTickLabels(ticks)
```

getLogTickLabels	<i>getLogTickLabels</i>
------------------	-------------------------

Description

Get ticklabels expressions for log scale plots

Usage

```
getLogTickLabels(ticks)
```

Arguments

ticks numeric values of the ticks

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- c(1, 5, 10, 50, 100, 500)
getLogTickLabels(ticks)
```

getPercentileTickLabels	<i>getPercentileTickLabels</i>
-------------------------	--------------------------------

Description

Get ticklabels expressions for percentiles of normal distribution scale plots

Usage

```
getPercentileTickLabels(ticks)
```

Arguments

ticks	numeric values of the ticks
-------	-----------------------------

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- rnorm(5)
getPercentileTickLabels(ticks)

# Get percentile of normal distribution
ticks <- qnorm(seq(1, 9) / 10)
getPercentileTickLabels(ticks)
```

getPiTickLabels	<i>getPiTickLabels</i>
-----------------	------------------------

Description

Get ticklabels expressions for plots with values as ratios of Pi

Usage

```
getPiTickLabels(ticks)
```

Arguments

ticks	numeric values of the ticks
-------	-----------------------------

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- seq(0, 2 * pi, pi / 2)
getPiTickLabels(ticks)
```

getPKRatioMeasure	<i>getPKRatioMeasure</i>
-------------------	--------------------------

Description

Get a summary table of PK Ratio within specific limits

Usage

```
getPKRatioMeasure(data, dataMapping = NULL, ratioLimits = c(1.5, 2))
```

Arguments

data	A data.frame to use for plot.
dataMapping	A PKRatioDataMapping object mapping x, y and aesthetic groups to their variable names of data.
ratioLimits	Numeric positive values limits

Value

A data.frame of summary of PK Ratio within specific limits

Examples

```
# Get summary of usual PK Ratio limits
pkData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

getPKRatioMeasure(data = pkData, dataMapping = PKRatioDataMapping$new(x = "x", y = "y"))

# Get summary of other PK Ratio limits
getPKRatioMeasure(
  data = pkData,
  dataMapping = PKRatioDataMapping$new(x = "x", y = "y"),
  ratioLimits = seq(1.5, 5, 0.5)
)
```

getSameLimits	<i>getSameLimits</i>
---------------	----------------------

Description

Get same limits from multiple sets of values

Usage

```
getSameLimits(...)
```

Arguments

... numeric values

Value

An array of 2 values

Examples

```
getSameLimits(seq(-3, 8), seq(-12, 4))
```

getSqrtTickLabels	<i>getSqrtTickLabels</i>
-------------------	--------------------------

Description

Get ticklabels expressions for sqrt scale plots

Usage

```
getSqrtTickLabels(ticks)
```

Arguments

ticks numeric values of the ticks

Value

Expressions to use in ticklabels input parameter of setXAxis and setYAxis functions

Examples

```
ticks <- sqrt(c(1, 5, 10, 50, 100, 500))
getSqrtTickLabels(ticks)
```

getSymmetricLimits	<i>getSymmetricLimits</i>
--------------------	---------------------------

Description

Get symmetric limits from a set of values

Usage

```
getSymmetricLimits(values)
```

Arguments

values	numeric values
--------	----------------

Value

An array of 2 symmetric values equally distant from 0

Examples

```
getSymmetricLimits(seq(-3, 8))
```

getTLFSettings	<i>getTLFSettings</i>
----------------	-----------------------

Description

Get Names of the default/global settings stored in tlfEnv. Can be used with getTLFSettings()

Usage

```
getTLFSettings(settingName)
```

Arguments

settingName	setting name as defined in enum tlfSettingsNames
-------------	--

Grouping

Grouping

Description

R6 class for mapping a group of variable(s) and their label to data

Public fields

`group` data.frame or character defining the groups or group variables to group by

`label` character printed name of the grouping

Methods

Public methods:

- [Grouping\\$new\(\)](#)
- [Grouping\\$getCaptions\(\)](#)
- [Grouping\\$clone\(\)](#)

Method `new()`: Create a new Grouping object

Usage:

```
Grouping$new(group, label = NULL)
```

Arguments:

`group` data.frame or character vector of groups

`label` character name of the group

Returns: A new Grouping object

Method `getCaptions()`: Get the caption associated to each group

Usage:

```
Grouping$getCaptions(data, metaData = NULL)
```

Arguments:

`data` data.frame to map

`metaData` list of information on the data

Returns: A vector of characters containing the captions associated to each group of data

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

GroupMapping

GroupMapping

Description

R6 class for mapping Grouping variables to data

Public fields

color R6 class Grouping object
fill R6 class Grouping object
linetype R6 class Grouping object
shape R6 class Grouping object
size R6 class Grouping object

Methods**Public methods:**

- [GroupMapping\\$new\(\)](#)
- [GroupMapping\\$clone\(\)](#)

Method new(): Create a new GroupMapping object

Usage:

```
GroupMapping$new(
  color = NULL,
  fill = NULL,
  linetype = NULL,
  shape = NULL,
  size = NULL
)
```

Arguments:

color R6 class Grouping object or its input
fill R6 class Grouping object or its input
linetype R6 class Grouping object or its input
shape R6 class Grouping object or its input
size R6 class Grouping object or its input

Returns: A new GroupMapping object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

HistogramDataMapping *HistogramDataMapping*

Description

R6 class for mapping x, bins, binwidth, stack and distribution to data

Super classes

```
tlf::XYDataMapping -> tlf::XYGDataMapping -> HistogramDataMapping
```

Public fields

`frequency` logical defining if histogram displays a frequency in y axis

`stack` logical defining if histogram bars should be stacked

`bins` number of bins or binning values/methods passed on `ggplot2::geom_histogram`

`binwidth` width of bins passed on `ggplot2::geom_histogram`. Overwrites bins

`distribution` Name of distribution to fit to the data. Only 2 distributions are currently available: "normal" and "logNormal"

Methods

Public methods:

- [HistogramDataMapping\\$new\(\)](#)
- [HistogramDataMapping\\$clone\(\)](#)

Method `new()`: Create a new HistogramDataMapping object

Usage:

```
HistogramDataMapping$new(
  frequency = FALSE,
  stack = FALSE,
  bins = NULL,
  binwidth = NULL,
  distribution = NULL,
  ...
)
```

Arguments:

frequency logical defining if histogram displays a frequency in y axis

stack logical defining if histogram bars should be stacked

bins argument passed on `ggplot2::geom_histogram`

binwidth width of bins passed on `ggplot2::geom_histogram`. Overwrites bins

distribution Name of distribution to fit to the data. Only 2 distributions are currently available: "normal" and "logNormal"

... parameters inherited from `XYGDataMapping`

Returns: A new `HistogramDataMapping` object

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other `DataMapping` classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

HistogramPlotConfiguration

HistogramPlotConfiguration

Description

R6 class defining the configuration of a `ggplot` object for histograms

Super class

```
tlf::PlotConfiguration -> HistogramPlotConfiguration
```

Methods

Public methods:

- [HistogramPlotConfiguration\\$clone\(\)](#)

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

HorizontalJustification

HorizontalJustification

Description

List of all available horizontal justifications for plot annotation text.

Usage

```
HorizontalJustification
```

Format

An object of class `list` of length 3.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

initializePlot	<i>initializePlot</i>
----------------	-----------------------

Description

Initialize a ggplot object and set its labels, grid, background and watermark

Usage

```
initializePlot(plotConfiguration = NULL)
```

Arguments

plotConfiguration
An optional PlotConfiguration object defining labels, grid, background and watermark

Value

A ggplot graphical object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/atom-plots.html>

See Also

Other atom plots: [addErrorbar\(\)](#), [addLine\(\)](#), [addRibbon\(\)](#), [addScatter\(\)](#)

Examples

```
# Initialize an empty plot
p <- initializePlot()

# Implement a customized configuration using PlotConfiguration
config <- PlotConfiguration$new(title = "My Plot", xlabel = "x variable", ylabel = "y variable")
p <- initializePlot(config)
```

isBetween	<i>isBetween</i>
-----------	------------------

Description

Assess if x is between left and right bounds. Shortcut for `x >= left & x <= right` if `strict=FALSE` (default). Shortcut for `x > left & x < right` if `strict=TRUE`.

Usage

```
isBetween(x, left, right, strict = FALSE)
```

Arguments

- | | |
|--------|---|
| x | Numeric values to assess |
| left | Numeric value(s) used as lower bound |
| right | Numeric value(s) used as upper bound |
| strict | Logical value defining if x is strictly between left and right. Default value is FALSE. |

Value

Logical values

Examples

```
isBetween(1:12, 7, 9)

x <- rnorm(1e2)
x[isBetween(x, -1, 1)]

isBetween(x, cos(x) + 1, cos(x) - 1)
```

Label	<i>Label</i>
-------	--------------

Description

R6 class defining text and font of labels

Active bindings

- | | |
|------|-----------------------------|
| text | character text of the label |
| font | Font object |

Methods

Public methods:

- [Label\\$new\(\)](#)
- [Label\\$createPlotTextBoxFont\(\)](#)
- [Label\\$createPlotTextFont\(\)](#)
- [Label\\$clone\(\)](#)

Method `new()`: Create a new Label object.

Usage:

```
Label$new(
  text = "",
  font = NULL,
  color = NULL,
  size = NULL,
  fontFace = NULL,
  fontFamily = NULL,
  angle = NULL,
  align = NULL,
  maxWidth = NULL,
  margin = NULL
)
```

Arguments:

`text` character text of the label

`font` Font object defining the font of the label

`color` character defining the color of the label

`size` numeric defining the size of the label

`fontFace` character defining the font face of the label as defined in helper enum `FontFaces`.

`fontFamily` character defining the font family of the label

`angle` numeric defining the angle of the label.

`align` character defining the alignment of the label as defined in helper enum `Alignments`.

`maxWidth` numeric that will be converted to a `ggplot2::unit` object (in "pt" unit) defining the maximum width of text box.

`margin` a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: A new Label object

Method `createPlotTextBoxFont()`: Create a `ggtext::element_textbox` directly convertible by `ggplot2::theme()`.

Usage:

```
Label$createPlotTextBoxFont(
  color = NULL,
  size = NULL,
  fontFace = NULL,
  fontFamily = NULL,
```

```

    angle = NULL,
    align = NULL,
    maxWidth = NULL,
    margin = NULL
  )

```

Arguments:

color character defining the color of the label
 size numeric defining the size of the label
 fontFace character defining the font face of the label as defined in helper enum FontFaces.
 fontFamily character defining the font family of the label
 angle numeric defining the angle of the label.
 align character defining the alignment of the label as defined in helper enum Alignments.
 maxWidth numeric that will be converted to a ggplot2::unit object (in "pt" unit) defining the maximum width of text box.
 margin a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: An element_text or element_blankobject.

Method createPlotTextFont(): Create a ggplot2::element_text() directly convertible by ggplot2::theme().

Usage:

```

Label$createPlotTextFont(
  color = NULL,
  size = NULL,
  fontFace = NULL,
  fontFamily = NULL,
  angle = NULL,
  align = NULL,
  margin = NULL
)

```

Arguments:

color character defining the color of the label
 size numeric defining the size of the label
 fontFace character defining the font face of the label as defined in helper enum FontFaces.
 fontFamily character defining the font family of the label
 angle numeric defining the angle of the label.
 align character defining the alignment of the label as defined in helper enum Alignments.
 margin a numeric vector of length 4 defining the size of the area (in pt) around the text in the followin order: top, right, bottom, left.

Returns: An element_text or element_blankobject.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```

Label$clone(deep = FALSE)

```

Arguments:

deep Whether to make a deep clone.

LabelConfiguration	<i>LabelConfiguration</i>
--------------------	---------------------------

Description

R6 class defining the configuration of the labels of a ggplot object

Active bindings

title Label object defining the title of the plot
subtitle Label object defining the subtitle of the plot
xlabel Label object defining the xlabel of the plot
ylabel Label object defining the ylabel of the plot
caption Label object defining the caption of the plot
y2label Label object defining the y2label of the plot

Methods

Public methods:

- `LabelConfiguration$new()`
- `LabelConfiguration$updatePlot()`
- `LabelConfiguration$clone()`

Method `new()`: Create a new LabelConfiguration object

Usage:

```
LabelConfiguration$new(  
  title = NULL,  
  subtitle = NULL,  
  xlabel = NULL,  
  ylabel = NULL,  
  caption = NULL  
)
```

Arguments:

title character or Label object defining title
subtitle character or Label object defining subtitle
xlabel character or Label object defining xlabel
ylabel character or Label object defining ylabel
caption character or Label object defining caption

Returns: A new LabelConfiguration object

Method `updatePlot()`: Update labels of a ggplot object and their properties

Usage:

```
LabelConfiguration$updatePlot(plotObject)
```

Arguments:

plotObject a ggplot object

Returns: A ggplot object

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

LegendConfiguration	<i>LegendConfiguration</i>
---------------------	----------------------------

Description

R6 class defining the legend configuration of a ggplot object

Active bindings

caption of legend defined as data.frame with caption properties

position of legend as defined in Enum LegendPositions

font Font object defining the font of the legend

background Background object defining the background of the legend

title character defining title of the legend

Methods

Public methods:

- [LegendConfiguration\\$new\(\)](#)
- [LegendConfiguration\\$updatePlot\(\)](#)
- [LegendConfiguration\\$clone\(\)](#)

Method new(): Create a new LegendConfiguration object

Usage:

```

LegendConfiguration$new(
  position = NULL,
  caption = NULL,
  title = NULL,
  font = NULL,
  background = NULL
)

```

Arguments:

position position of the legend as defined by enum LegendPositions

caption data.frame containing the properties of the legend caption

title character or Label object defining the title of the legend. A value of NULL removes the title.

font Font object defining the font of the legend caption

background BackgroundElement object defining the background of the legend

Returns: A new LegendConfiguration object

Method updatePlot(): Update legend configuration on a ggplot object

Usage:

```
LegendConfiguration$updatePlot(plotObject)
```

Arguments:

plotObject ggplot object

Returns: A ggplot object with updated axis properties

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

LegendPositions	<i>LegendPositions</i>
-----------------	------------------------

Description

List of all available legend positions

Usage

LegendPositions

Format

An object of class list of length 17.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

LegendTypes	<i>LegendTypes</i>
-------------	--------------------

Description

List of all available legend types

Usage

LegendTypes

Format

An object of class list of length 5.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

LineElement

LineElement

Description

R6 class defining the properties of background line elements

Super class

`tlf::BackgroundElement -> LineElement`

Methods

Public methods:

- [LineElement\\$createPlotElement\(\)](#)
- [LineElement\\$clone\(\)](#)

Method `createPlotElement()`: Create a `ggplot2::element_line` directly usable by `ggplot2::theme`.

Usage:

```
LineElement$createPlotElement(color = NULL, size = NULL, linetype = NULL)
```

Arguments:

`color` character color of the frame of the background element

`size` character size of the frame of the background element

`linetype` character linetype of the frame of the background element

Returns: An `element_line` object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `PlotConfiguration` classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

Linetypes*Linetypes*

Description

Enum of ggplot2 linetypes

Usage

Linetypes

Format

An object of class list of length 7.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Examples

```
# Use ggplot2 to plot and label Linetypes
linesData <- data.frame(
  x = 0,
  y = seq_along(Linetypes),
  linetype = factor(names(Linetypes), levels = names(Linetypes))
)

ggplot2::ggplot(data = linesData) +
  ggplot2::theme_void() +
  ggplot2::geom_hline(ggplot2::aes(yintercept = y, linetype = linetype)) +
  # Add linetype names from enum below the displayed linetype
  ggplot2::geom_text(ggplot2::aes(x = x, y = y, label = linetype), nudge_y = -0.2, size = 4) +
  # Use scale to display the actual linetype
  ggplot2::scale_linetype_manual(values = as.character(unlist(Linetypes))) +
  # Remove the legend as the linetype name is labelled below the linetype
  ggplot2::guides(linetype = "none")

# Perform a line plot with blue long dashes as linetype
addLine(
  x = 1:10,
  y = rlnorm(10),
  linetype = Linetypes$longdash,
  color = "blue",
  size = 1
)
```

)

loadThemeFromJson	<i>loadThemeFromJson</i>
-------------------	--------------------------

Description

Load theme object from json file. A template of a json theme is available at `system.file(package="tlf", "theme-maker", "theme-template.json")`

Usage

`loadThemeFromJson(jsonFile)`

Arguments

`jsonFile` path of json file

Value

A Theme object

loadTLFSettings	<i>loadTLFSettings</i>
-----------------	------------------------

Description

Load TLF global settings from a file

Usage

`loadTLFSettings(file)`

Arguments

`file` .RData file containing the settings

mean-1.96sd	<i>mean-1.96sd</i>
-------------	--------------------

Description

Calculate mean-1.96SD

Usage

``mean-1.96sd`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to $\text{mean}(x) - 1.96 * \text{sd}(x)$

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate mean-1.96SD
`mean-1.96sd`(rnorm(1000))
```

mean-sd	<i>mean-sd</i>
---------	----------------

Description

Calculate mean-SD

Usage

``mean-sd`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to $\text{mean}(x) - \text{sd}(x)$

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate mean-SD
`mean-sd`(rnorm(1000))
```

<code>mean+1.96sd</code>	<i>mean+1.96sd</i>
--------------------------	--------------------

Description

Calculate $\text{mean} + 1.96\text{SD}$

Usage

```
`mean+1.96sd`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to $\text{mean}(x) + 1.96 * \text{sd}(x)$

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate mean+1.96SD
`mean+1.96sd`(rnorm(1000))
```

mean+sd

mean+sd

Description

Calculate mean+SD

Usage

```
`mean+sd`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to mean(x)+sd(x)

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate mean-SD
`mean+sd`(rnorm(1000))
```

median-1.5IQR	<i>median-1.5IQR</i>
---------------	----------------------

Description

Calculate median-1.5IQR

Usage

``median-1.5IQR`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `median(x)-1.5*iqr(x)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate median-1.5IQR
`median-1.5IQR`(rnorm(1000))
```

median-IQR	<i>median-IQR</i>
------------	-------------------

Description

Calculate median-IQR

Usage

``median-IQR`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `median(x)-iqr(x)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#)

Examples

```
# Calculate median-IQR
`median-IQR`(rnorm(1000))
```

median+1.5IQR	<i>median+1.5IQR</i>
---------------	----------------------

Description

Calculate `median+1.5IQR`

Usage

```
`median+1.5IQR`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `median(x)+1.5*iqr(x)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate median+1.5IQR
`median+1.5IQR`(rnorm(1000))
```

 median+IQR

median+IQR

Description

Calculate median+IQR

Usage

```
`median+IQR`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to median(x)+iqr(x)

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate median+IQR
`median+IQR`(rnorm(1000))
```

 MoleculePlots

MoleculePlots

Description

List of all available molecule plots

Usage

```
MoleculePlots
```

Format

An object of class list of length 15.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

ObservedDataMapping	<i>ObservedDataMapping</i>
---------------------	----------------------------

Description

R6 class for mapping x, y, of observed data for a time profile plot

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> ObservedDataMapping`

Public fields

`error` mapping error bars around scatter points
`mdv` mapping missing dependent variable
`ymin` mapping error bars around scatter points
`ymax` mapping error bars around scatter points
`y2Axis` Name of y2Axis variable to map
`lloq` mapping lloq lines

Methods**Public methods:**

- [ObservedDataMapping\\$new\(\)](#)
- [ObservedDataMapping\\$checkMapData\(\)](#)
- [ObservedDataMapping\\$requireDualAxis\(\)](#)
- [ObservedDataMapping\\$getLeftAxis\(\)](#)
- [ObservedDataMapping\\$getRightAxis\(\)](#)
- [ObservedDataMapping\\$clone\(\)](#)

Method `new()`: Create a new ObservedDataMapping object

Usage:

```
ObservedDataMapping$new(
  x,
  y,
  ymin = NULL,
  ymax = NULL,
  y2Axis = NULL,
  group = NULL,
  color = NULL,
  shape = NULL,
  error = NULL,
  uncertainty = lifecycle::deprecated(),
  mdv = NULL,
  data = NULL,
  lloq = NULL
)
```

Arguments:

x Name of x variable to map
 y Name of y variable to map
 ymin mapping lower end of error bars around scatter points
 ymax mapping upper end of error bars around scatter points
 y2Axis Name of y2Axis variable to map
 group R6 class Grouping object or its input
 color R6 class Grouping object or its input
 shape R6 class Grouping object or its input
 error mapping error bars around scatter points
 uncertainty **[Deprecated]** uncertainty were replaced by error argument. Mapping error bars around scatter points.
 mdv mapping missing dependent variable
 data data.frame to map used by .smartMapping
 lloq mapping lloq lines

Returns: A new ObservedDataMapping object

Method checkMapData(): Check that data variables include map variables

Usage:

```
ObservedDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check
 metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method requireDualAxis(): Assess if data require a dual axis plot

Usage:

```
ObservedDataMapping$requireDualAxis(data)
```

Arguments:

data data.frame to check

Returns: A logical

Method getLeftAxis(): Render NA values for all right axis data

Usage:

ObservedDataMapping\$getLeftAxis(data)

Arguments:

data A data.frame

Returns: A data.frame to be plotted in left axis

Method getRightAxis(): Render NA values for all left axis data

Usage:

ObservedDataMapping\$getRightAxis(data)

Arguments:

data A data.frame

Returns: A data.frame to be plotted in right axis

Method clone(): The objects of this class are cloneable with this method.

Usage:

ObservedDataMapping\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

ObsVsPredDataMapping *ObsVsPredDataMapping*

Description

Class for mapping variables in observations vs predictions plot

Super classes

t1f::XYDataMapping -> t1f::XYGDataMapping -> ObsVsPredDataMapping

Public fields

lines list of ratio limits to plot as horizontal lines
 xmin mapping of upper value of error bars around scatter points
 xmax mapping of lower value of error bars around scatter points
 smoother regression function name
 lloq mapping lloq lines

Methods**Public methods:**

- [ObsVsPredDataMapping\\$new\(\)](#)
- [ObsVsPredDataMapping\\$checkMapData\(\)](#)
- [ObsVsPredDataMapping\\$clone\(\)](#)

Method new(): Create a new ObsVsPredDataMapping object

Usage:

```
ObsVsPredDataMapping$new(
  x = NULL,
  y = NULL,
  xmin = NULL,
  xmax = NULL,
  lines = DefaultDataMappingValues$obsVsPred,
  smoother = NULL,
  lloq = NULL,
  ...
)
```

Arguments:

x Name of x variable to map
 y Name of y variable to map
 xmin mapping of upper value of error bars around scatter points
 xmax mapping of lower value of error bars around scatter points
 lines list of lines to plot
 smoother smoother function or parameter
 lloq mapping lloq lines To map a loess smoother to the plot, use smoother="loess"
 ... parameters inherited from XYGDataMapping

Returns: A new ObsVsPredDataMapping object

Method checkMapData(): Check that data variables include map variables

Usage:

```
ObsVsPredDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check
 metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

ObsVsPredPlotConfiguration

ObsVsPredPlotConfiguration

Description

R6 class defining the configuration of a ggplot object for Obs vs Pred plots

Super class

```
tlf::PlotConfiguration -> ObsVsPredPlotConfiguration
```

Public fields

defaultSymmetricAxes Default option setting symmetric xAxis and/or yAxis limits when creating a ObsVsPredPlotConfiguration object

lloqDirection Whether to draw LLOQ lines for x (vertical), y (horizontal) or x and y (both).

foldLinesLegend Whether to draw fold lines in legend. default to FALSE.

foldLinesLegendDiagonal Whether to draw diagonal lines in legend for fold lines. default to FALSE.

Active bindings

foldLineslegendType translation of foldLinesLegendDiagonal in geom type.

Methods

Public methods:

- [ObsVsPredPlotConfiguration\\$new\(\)](#)
- [ObsVsPredPlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new `ObsVsPredPlotConfiguration` object

Usage:

```
ObsVsPredPlotConfiguration$new(  
  lloqDirection = "vertical",  
  foldLinesLegend = FALSE,  
  foldLinesLegendDiagonal = FALSE,  
  ...  
)
```

Arguments:

`lloqDirection` Whether to draw LLOQ lines for x (vertical), y (horizontal) or x and y (both).
`foldLinesLegend` Whether to draw fold lines in legend. default to FALSE.
`foldLinesLegendDiagonal` Whether to draw diagonal lines in legend for fold lines. default to FALSE.
... parameters inherited from `PlotConfiguration`

Returns: A new `CumulativeTimeProfilePlotConfiguration` object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `PlotConfiguration` classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

Percentile0%	<i>Percentile0%</i>
--------------	---------------------

Description

Calculate `Percentile0%` i.e. min value

Usage

``Percentile0%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 0)`

See Also

Other stat functions: [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile0%
`Percentile0%`(rnorm(1000))
```

Percentile1%	<i>Percentile1%</i>
--------------	---------------------

Description

Calculate Percentile1%

Usage

``Percentile1%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 1/100)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile1%
`Percentile1%`(rnorm(1000))
```

Percentile10%	<i>Percentile10%</i>
---------------	----------------------

Description

Calculate Percentile10%

Usage

```
`Percentile10%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 10/100)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile10%
`Percentile10%`(rnorm(1000))
```

Percentile100%	<i>Percentile100%</i>
----------------	-----------------------

Description

Calculate Percentile100% i.e. max value

Usage

``Percentile100%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 1)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile100%
`Percentile100%`(rnorm(1000))
```

Percentile15%	<i>Percentile15%</i>
---------------	----------------------

Description

Calculate Percentile15%

Usage

``Percentile15%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 15/100)`

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile1%](#)(), [Percentile2.5%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#)(), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile95%](#)(), [Percentile97.5%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile15%
`Percentile15%`(rnorm(1000))
```

Percentile2.5%	<i>Percentile2.5%</i>
----------------	-----------------------

Description

Calculate Percentile2.5%

Usage

```
`Percentile2.5%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 2.5/100)`

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile15%](#)(), [Percentile1%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#)(), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile95%](#)(), [Percentile97.5%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile2.5%
`Percentile2.5`(rnorm(1000))
```

Percentile20%	<i>Percentile20%</i>
---------------	----------------------

Description

Calculate Percentile20%

Usage

```
`Percentile20%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 20/100)`

See Also

Other stat functions: [Percentile0%](#), [Percentile100%](#), [Percentile10%](#), [Percentile15%](#), [Percentile1%](#), [Percentile2.5%](#), [Percentile25%](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#), [Percentile5%](#), [Percentile75%](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#), [Percentile85%](#), [Percentile90%](#), [Percentile95%](#), [Percentile97.5%](#), [Percentile99%](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile20%
`Percentile20%`(rnorm(1000))
```

Percentile25%-1.5IQR	<i>Percentile25%-1.5IQR</i>
----------------------	-----------------------------

Description

Calculate Percentile25%-1.5IQR

Usage

``Percentile25%-1.5IQR`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 0.25)-1.5*iqr(x)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile25%-1.5IQR
`Percentile25%-1.5IQR`(rnorm(1000))
```

Percentile25%	<i>Percentile25%</i>
---------------	----------------------

Description

Calculate Percentile25% i.e. 1st quartile value

Usage

``Percentile25%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 25/100)`

See Also

Other stat functions: [Percentile0%](#), [Percentile100%](#), [Percentile10%](#), [Percentile15%](#), [Percentile1%](#), [Percentile2.5%](#), [Percentile20%](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#), [Percentile5%](#), [Percentile75%](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#), [Percentile85%](#), [Percentile90%](#), [Percentile95%](#), [Percentile97.5%](#), [Percentile99%](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile25%
`Percentile25%`(rnorm(1000))
```

Percentile5%	<i>Percentile5%</i>
--------------	---------------------

Description

Calculate Percentile5%

Usage

```
`Percentile5%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 5/100)`

See Also

Other stat functions: [Percentile0%](#), [Percentile100%](#), [Percentile10%](#), [Percentile15%](#), [Percentile1%](#), [Percentile2.5%](#), [Percentile20%](#), [Percentile25%](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#), [Percentile75%](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#), [Percentile85%](#), [Percentile90%](#), [Percentile95%](#), [Percentile97.5%](#), [Percentile99%](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile5%  
`Percentile5%`(rnorm(1000))
```

Percentile50%	<i>Percentile50%</i>
---------------	----------------------

Description

Calculate Percentile50% i.e. median value

Usage

```
`Percentile50%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 50/100)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile50%  
`Percentile50%`(rnorm(1000))
```

Percentile75%+1.5IQR	<i>Percentile75%+1.5IQR</i>
----------------------	-----------------------------

Description

Calculate Percentile75%+1.5IQR

Usage

``Percentile75%+1.5IQR`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 0.75)+1.5*iqr(x)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile75%+1.5IQR
`Percentile75%+1.5IQR`(rnorm(1000))
```

Percentile75%	<i>Percentile75%</i>
---------------	----------------------

Description

Calculate Percentile75% i.e. 3rd quartile value

Usage

``Percentile75%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to quantile(x, 75/100)

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile15%](#)(), [Percentile1%](#)(), [Percentile2.5%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#)(), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile95%](#)(), [Percentile97.5%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile75%
`Percentile75%`(rnorm(1000))
```

Percentile80%	<i>Percentile80%</i>
---------------	----------------------

Description

Calculate Percentile80%

Usage

```
`Percentile80%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to quantile(x, 80/100)

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile15%](#)(), [Percentile1%](#)(), [Percentile2.5%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile95%](#)(), [Percentile97.5%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile80%
`Percentile80%`(rnorm(1000))
```

Percentile85%	<i>Percentile85%</i>
---------------	----------------------

Description

Calculate Percentile85%

Usage

```
`Percentile85%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 85/100)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile85%
`Percentile85%`(rnorm(1000))
```

Percentile90%	Percentile90%
---------------	---------------

Description

Calculate Percentile90%

Usage

``Percentile90%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to `quantile(x, 90/100)`

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [Percentile99%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile90%
`Percentile90%`(rnorm(1000))
```

Percentile95%	Percentile95%
---------------	---------------

Description

Calculate Percentile95%

Usage

``Percentile95%`(x)`

Arguments

x Numeric values

Value

Numeric value corresponding to quantile(x, 95/100)

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile15%](#)(), [Percentile1%](#)(), [Percentile2.5%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#)(), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile97.5%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile95%
`Percentile95%`(rnorm(1000))
```

Percentile97.5%	<i>Percentile97.5%</i>
-----------------	------------------------

Description

Calculate Percentile97.5%

Usage

```
`Percentile97.5%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to quantile(x, 97.5/100)

See Also

Other stat functions: [Percentile0%](#)(), [Percentile100%](#)(), [Percentile10%](#)(), [Percentile15%](#)(), [Percentile1%](#)(), [Percentile2.5%](#)(), [Percentile20%](#)(), [Percentile25%](#)(), [Percentile25%-1.5IQR\(\)](#), [Percentile50%](#)(), [Percentile5%](#)(), [Percentile75%](#)(), [Percentile75%+1.5IQR\(\)](#), [Percentile80%](#)(), [Percentile85%](#)(), [Percentile90%](#)(), [Percentile95%](#)(), [Percentile99%](#)(), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile97.5%
`Percentile97.5`(rnorm(1000))
```

Percentile99%	<i>Percentile99%</i>
---------------	----------------------

Description

Calculate Percentile99%

Usage

```
`Percentile99%`(x)
```

Arguments

x Numeric values

Value

Numeric value corresponding to quantile(x, 99/100)

See Also

Other stat functions: [Percentile0%\(\)](#), [Percentile100%\(\)](#), [Percentile10%\(\)](#), [Percentile15%\(\)](#), [Percentile1%\(\)](#), [Percentile2.5%\(\)](#), [Percentile20%\(\)](#), [Percentile25%\(\)](#), [Percentile25%-1.5IQR\(\)](#), [Percentile50%\(\)](#), [Percentile5%\(\)](#), [Percentile75%\(\)](#), [Percentile75%+1.5IQR\(\)](#), [Percentile80%\(\)](#), [Percentile85%\(\)](#), [Percentile90%\(\)](#), [Percentile95%\(\)](#), [Percentile97.5%\(\)](#), [mean+1.96sd\(\)](#), [mean+sd\(\)](#), [mean-1.96sd\(\)](#), [mean-sd\(\)](#), [median+1.5IQR\(\)](#), [median+IQR\(\)](#), [median-1.5IQR\(\)](#), [median-IQR\(\)](#)

Examples

```
# Calculate Percentile99%
`Percentile99%`(rnorm(1000))
```

PieChartDataMapping	<i>PieChartDataMapping</i>
---------------------	----------------------------

Description

R6 class for mapping x, y, and group to data

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> PieChartDataMapping`

Methods**Public methods:**

- [PieChartDataMapping\\$clone\(\)](#)

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

Usage:

`PieChartDataMapping$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `DataMapping` classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

PieChartPlotConfiguration	<i>PieChartPlotConfiguration</i>
---------------------------	----------------------------------

Description

R6 class defining the configuration of a ggplot object for pie charts

Super class

`tlf::PlotConfiguration -> PieChartPlotConfiguration`

Public fields

colorPalette color palette property from ggplot2
 chartFont Font object defining properties of text within pie chart
 start Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction
 clockwiseDirection logical defining if values are displayed in clockwise order

Methods**Public methods:**

- [PieChartPlotConfiguration\\$new\(\)](#)
- [PieChartPlotConfiguration\\$clone\(\)](#)

Method new(): Create a new PieChartPlotConfiguration object

Usage:

```
PieChartPlotConfiguration$new(
  colorPalette = NULL,
  chartFont = NULL,
  start = 0,
  clockwiseDirection = TRUE,
  ...
)
```

Arguments:

colorPalette color palette property from ggplot2
 chartFont Font object defining properties of text within pie chart
 start Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction
 clockwiseDirection logical defining if values are displayed in clockwise order
 ... parameters inherited from PlotConfiguration

Returns: A new PieChartPlotConfiguration object

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

PKRatioDataMapping	<i>PKRatioDataMapping</i>
--------------------	---------------------------

Description

R6 class for mapping x, y, GroupMapping and pkRatio lines variables to data

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> PKRatioDataMapping`

Public fields

`lines` list of ratio limits to plot as horizontal lines

`ymin` mapping of upper value of error bars around scatter points

`ymax` mapping of lower value of error bars around scatter points

Methods**Public methods:**

- `PKRatioDataMapping$new()`
- `PKRatioDataMapping$checkMapData()`
- `PKRatioDataMapping$clone()`

Method `new()`: Create a new PKRatioDataMapping object

Usage:

```
PKRatioDataMapping$new(
  x = NULL,
  y = NULL,
  ymin = NULL,
  ymax = NULL,
  lines = DefaultDataMappingValues$pkRatio,
  ...
)
```

Arguments:

`x` Name of x variable to map

`y` Name of y variable to map

`ymin` mapping of upper value of error bars around scatter points

`ymax` mapping of lower value of error bars around scatter points

`lines` List of ratio limits to display as horizontal lines

`...` parameters inherited from XYGDataMapping

Returns: A new PKRatioDataMapping object

Method `checkMapData()`: Check that data variables include map variables

Usage:

```
PKRatioDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check

metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

PKRatioPlotConfiguration

PKRatioPlotConfiguration

Description

R6 class defining the configuration of a ggplot object for PK ratio plots

Super class

```
tlf::PlotConfiguration -> PKRatioPlotConfiguration
```

Public fields

defaultYScale Default yAxis scale value when creating a PKRatioPlotConfiguration object

Methods**Public methods:**

- [PKRatioPlotConfiguration\\$clone\(\)](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

PlotAnnotationTextSize

PlotAnnotationTextSize

Description

List of default text sizes for plot annotations.

Usage

PlotAnnotationTextSize

Format

An object of class list of length 12.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

plotBoxWhisker

plotBoxWhisker

Description

Producing box-and-whisker plots

Usage

```
plotBoxWhisker(
  data,
  metaData = NULL,
  outliers = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>outliers</code>	Logical defining if outliers should be included in boxplot
<code>dataMapping</code>	A BoxWhiskerDataMapping object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional BoxWhiskerConfiguration object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/box-whisker-vignette.html>

See Also

Other molecule plots: [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce box-and-whisker plots of log-normal distributed data
boxData <- data.frame(x = c(rep("A", 500), rep("B", 500)), y = rlnorm(1000))

plotBoxWhisker(data = boxData, dataMapping = BoxWhiskerDataMapping$new(x = "x", y = "y"))

# Remove outliers from boxplot
plotBoxWhisker(
  data = boxData,
```

```

    dataMapping = BoxWhiskerDataMapping$new(x = "x", y = "y"),
    outliers = FALSE
  )

```

PlotConfiguration	<i>PlotConfiguration</i>
-------------------	--------------------------

Description

R6 class defining the configuration of a ggplot object

Public fields

export R6 class ExportConfiguration defining properties for saving/exporting plot
 defaultXScale Default xAxis scale value when creating a PlotConfiguration object
 defaultYScale Default yAxis scale value when creating a PlotConfiguration object
 defaultExpand Default expand value when creating a PlotConfiguration object
 defaultSymmetricAxes Default option setting symmetric xAxis and/or yAxis limits when creating a PlotConfiguration object

Active bindings

labels LabelConfiguration object defining properties of labels
 legend LegendConfiguration object defining properties of legend
 xAxis XAxisConfiguration object defining properties of x-axis
 yAxis YAxisConfiguration object defining properties of y-axis
 background BackgroundConfiguration object defining properties of background
 lines ThemeAestheticSelections defining properties of lines
 ribbons ThemeAestheticSelections defining properties of ribbons
 points ThemeAestheticSelections defining properties of points
 errorbars ThemeAestheticSelections defining properties of error bars

Methods

Public methods:

- [PlotConfiguration\\$new\(\)](#)
- [PlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new PlotConfiguration object

Usage:

```
PlotConfiguration$new(  
  title = NULL,  
  subtitle = NULL,  
  xlabel = NULL,  
  ylabel = NULL,  
  caption = NULL,  
  legend = NULL,  
  legendTitle = NULL,  
  legendPosition = NULL,  
  xAxis = NULL,  
  xScale = NULL,  
  xValuesLimits = NULL,  
  xAxisLimits = NULL,  
  xLimits = lifecycle::deprecated(),  
  yAxis = NULL,  
  yScale = NULL,  
  yValuesLimits = NULL,  
  yAxisLimits = NULL,  
  yLimits = lifecycle::deprecated(),  
  background = NULL,  
  plotArea = NULL,  
  panelArea = NULL,  
  xGrid = NULL,  
  yGrid = NULL,  
  watermark = NULL,  
  lines = NULL,  
  points = NULL,  
  ribbons = NULL,  
  errorbars = NULL,  
  export = NULL,  
  name = NULL,  
  format = NULL,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL,  
  data = NULL,  
  metaData = NULL,  
  dataMapping = NULL  
)
```

Arguments:

title character or Label object defining plot title
subtitle character or Label object defining plot subtitle
xlabel character or Label object defining plot xlabel
ylabel character or Label object defining plot ylabel
caption character or Label object defining plot caption
legend LegendConfiguration object defining legend properties

legendTitle character or Label object defining legend title
 legendPosition character defining legend position. Use Enum LegendPositions to get a list of available to legend positions.
 xAxis XAxisConfiguration object defining x-axis properties
 xScale name of X-axis scale. Use enum Scaling to access predefined scales.
 xValuesLimits numeric vector of length 2 defining x values limits
 xAxisLimits numeric vector of length 2 defining x-axis limits
 xLimits **[Deprecated]**. Replaced by xAxisLimits argument.
 yAxis YAxisConfiguration object defining y-axis properties
 yScale name of y-axis scale. Use enum Scaling to access predefined scales.
 yValuesLimits numeric vector of length 2 defining x values limits
 yAxisLimits numeric vector of length 2 defining x-axis limits
 yLimits **[Deprecated]**. Replaced by yAxisLimits argument.
 background BackgroundConfiguration object defining background properties
 plotArea BackgroundElement object defining properties of plot area
 panelArea BackgroundElement object defining properties of panel area
 xGrid LineElement object defining properties of x-grid background
 yGrid LineElement object defining properties of y-grid background
 watermark character or Label object defining watermark
 lines ThemeAestheticSelections object or list defining how lines are plotted
 points ThemeAestheticSelections object or list defining how points are plotted
 ribbons ThemeAestheticSelections object or list defining how ribbons are plotted
 errorbars ThemeAestheticSelections object or list defining how errorbars are plotted
 export R6 class ExportConfiguration defining properties for saving/exporting plot
 name character defining the name of the file to be saved (without extension)
 format character defining the format of the file to be saved.
 width numeric values defining the width in units of the plot dimensions after saving
 height numeric values defining the height in units of the plot dimensions after saving
 units character defining the unit of the saving dimension
 dpi numeric value defining plot resolution (dots per inch)
 data data.frame used by .smartMapping
 metaData list of information on data
 dataMapping R6 class or subclass XYDataMapping

Returns: A new PlotConfiguration object

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/plot-configuration.html>

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

PlotConfigurations	<i>PlotConfigurations</i>
--------------------	---------------------------

Description

List of all available molecule plots

Usage

PlotConfigurations

Format

An object of class list of length 23.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

plotCumulativeTimeProfile
plotCumulativeTimeProfile

Description

Producing Cumulative Time Profile plots

Usage

```
plotCumulativeTimeProfile(  
  data = NULL,  
  metaData = NULL,  
  dataMapping = NULL,  
  colorPalette = NULL,  
  plotConfiguration = NULL,  
  plotObject = NULL  
)
```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
dataMapping	A CumulativeTimeProfileDataMapping object mapping x and aesthetic groups to their variable names of data.
colorPalette	Optional character values defining a ggplot2 colorPalette (e.g. "Set1" or "Spectral")
plotConfiguration	An optional CumulativeTimeProfilePlotConfiguration object defining labels, grid, background and watermark.
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Define data to be plotted as cumulative time profile
time <- seq(1, 10)
data <- data.frame(
  x = rep(time),
  y = c(exp(-time / 10), 1 - exp(-time / 10)),
  legend = rep(c("decreasing area", "increasing area"), each = 10)
)

# Produce a Cumulative Time Profile plot
plotCumulativeTimeProfile(
  data = data,
  dataMapping = CumulativeTimeProfileDataMapping$new(x = "x", y = "y", fill = "legend")
)

# Produce a Cumulative Time Profile plot with a ggplot2 color palette
plotCumulativeTimeProfile(
  data = data,
  dataMapping = CumulativeTimeProfileDataMapping$new(
    x = "x",
    y = "y",
    fill = "legend"
  ),
  colorPalette = "Set1"
)
```

plotDDIRatio

plotDDIRatio

Description

Producing DDI Ratio plots

Usage

```
plotDDIRatio(
  data,
  metaData = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  residualsVsObserved = NULL,
  foldDistance = NULL,
  deltaGuest = NULL,
  plotObject = NULL
)
```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
dataMapping	A DDIRatioDataMapping object mapping x, y and aesthetic groups to their variable names of data.
plotConfiguration	An optional DDIRatioPlotConfiguration object defining labels, grid, background and watermark.
residualsVsObserved	Optional logical value defining if DDI Ratio plot is drawn as residuals vs observed, instead of predicted vs observed.
foldDistance	Numeric values of fold distance lines to display in log plots. This argument is internally translated into lines field of dataMapping. Caution: this argument is meant for log scaled plots and since fold distance is a ratio it is expected positive. In particular, line of identity corresponds to a foldDistance of 1.
deltaGuest	Numeric value parameter of Guest function
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/pk-ratio-vignette.html>

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce DDI Ratio plot
ddiData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

plotDDIRatio(data = ddiData, dataMapping = DDIRatioDataMapping$new(x = "x", y = "y"))

# Produce DDI Ratio plot with user-defined horizontal lines
plotDDIRatio(
  data = ddiData,
  dataMapping = DDIRatioDataMapping$new(x = "x", y = "y"),
  foldDistance = c(1, 10),
  deltaGuest = 1.25,
  residualsVsObserved = TRUE
```

)

plotGrid

*Create a plot grid***Description**

Create a plot grid using the `patchwork::wrap_plots()` function. The required arguments are supplied through the `PlotGridConfiguration` object.

Usage

```
plotGrid(plotGridConfiguration)
```

Arguments

`plotGridConfiguration`

A `PlotGridConfiguration` object, which is an R6 class object that defines properties of a plot grid (like number of rows, columns, labels, etc.).

References

For more, see: <https://patchwork.data-imaginist.com/articles/patchwork.html>

See Also

Other molecule plots: `plotBoxWhisker()`, `plotCumulativeTimeProfile()`, `plotDDIRatio()`, `plotHistogram()`, `plotObsVsPred()`, `plotObservedTimeProfile()`, `plotPKRatio()`, `plotPieChart()`, `plotQQ()`, `plotResVsPred()`, `plotResVsTime()`, `plotSimulatedTimeProfile()`, `plotTimeProfile()`, `plotTornado()`

Examples

```
library(ggplot2)
library(tlfr)

# only `{tlfr}` -----

# plots to be arranged in a grid
set.seed(123)
ls_plots <- list(
  plotHistogram(x = rnorm(100)),
  plotHistogram(x = rnorm(100, mean = 3)),
  plotHistogram(x = rnorm(100, mean = 10))
)

# create an instance of plot configuration class
plotGridObj <- PlotGridConfiguration$new(plotList = ls_plots)
```

```

# specify further customizations for the plot grid
plotGridObj$title <- "my combined plot"
plotGridObj$subtitle <- "something clever"
plotGridObj$caption <- "my sources"
plotGridObj$nColumns <- 2L
plotGridObj$tagLevels <- "A"
plotGridObj$tagPrefix <- "Plot ("
plotGridObj$tagSuffix <- ")"
plotGridObj$tagColor <- "blue"
plotGridObj$tagSize <- 15
plotGridObj$tagAngle <- 45
plotGridObj$tagPosition <- TagPositions$top
plotGridObj$titleHorizontalJustification <- HorizontalJustification$middle
plotGridObj$subtitleHorizontalJustification <- HorizontalJustification$middle

# plot the grid
plotGrid(plotGridObj)

# `{tlf}` and `{ggplot2}` -----

# `{tlf}` plot
set.seed(123)
p1 <- plotBoxWhisker(mtcars,
  dataMapping = BoxWhiskerDataMapping$new(x = "am", y = "wt"), outliers = FALSE
)

# custom `{ggplot2}` plot
set.seed(123)
p2 <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()

# create an instance of plot configuration class
plotGridObj2 <- PlotGridConfiguration$new(list(p1, p2))

# specify further customizations for the plot grid
plotGridObj2$nColumns <- 1L
plotGridObj2$tagLevels <- "i"

# plot the grid
plotGrid(plotGridObj2)

```

PlotGridConfiguration *Class for creating a plot grid*

Description

An R6 class defining the configuration for `{patchwork}` plot grid used to create a grid of plots from `{tlf}`. It holds values for all relevant plot properties.

Value

A PlotGridConfiguration object.

Customizing

You can change the default values present in public fields.

For example, if you want to specify a new position for tags, you will have to do the following:

```
myPlotGridConfiguration <- PlotGridConfiguration$new()
myPlotGridConfiguration$tagPosition <- TagPositions$right
```

For more, see examples.

Specifying fonts

A font is a particular set of glyphs (character shapes), differentiated from other fonts in the same family by additional properties such as stroke weight, slant, relative width, etc.

A font face (aka typeface) is the design of lettering, characterized by variations in size, weight (e.g. bold), slope (e.g. italic), width (e.g. condensed), and so on. The available font faces can be seen using FontFaces list.

A font family is a grouping of fonts defined by shared design styles.

The available font families will depend on which fonts have been installed on your computer. This information can be extracted by running the following code:

```
# install.packages("systemfonts")
library(systemfonts)
system_fonts()
```

Saving plot

By default, the plots will be shown in plot pane of your IDE, but the plots can also be saved to a file using the ggplot2::ggsave() function.

```
myPlot <- plotGrid(...)
ggplot2::ggsave(filename = "plot_1.png", plot = myPlot)
```

Super class

`ospsuite.utils::Printable` -> PlotGridConfiguration

Public fields

`plotList` A list containing ggplot objects.

`title`, `subtitle`, `caption` Text strings to use for the various plot annotations, where `plot` refers to the grid of plots as a whole.

`titleColor`, `titleSize`, `titleFontFace`, `titleFontFamily`, `titleHorizontalJustification`, `titleVerticalJustification` Aesthetic properties for the plot title.

`subtitleColor`, `subtitleSize`, `subtitleFontFace`, `subtitleFontFamily`, `subtitleHorizontalJustification`, `subtitleVerticalJustification` Aesthetic properties for the plot subtitle.

`captionColor`, `captionSize`, `captionFontFace`, `captionFontFamily`, `captionHorizontalJustification`, `captionVerticalJustification` Aesthetic properties for the plot caption.

`tagLevels` A character vector defining the enumeration format to use at each level. Possible values are 'a' for lowercase letters, 'A' for uppercase letters, '1' for numbers, 'i' for lowercase Roman numerals, and 'I' for uppercase Roman numerals. It can also be a list containing character vectors defining arbitrary tag sequences. If any element in the list is a scalar and one of 'a', 'A', '1', 'i', or 'I', this level will be expanded to the expected sequence.

`tagPrefix`, `tagSuffix` Strings that should appear before or after the tag.

`tagSeparator` A separator between different tag levels.

`tagPosition` Position of the tag for an individual plot with respect to that plot. Default is topleft. For all available options, see `TagPositions`.

`tagColor`, `tagSize`, `tagFontFamily`, `tagFontFace`, `tagHorizontalJustification`, `tagVerticalJustification`, `tagAngle` Aesthetic properties of individual plot tag text. For more detailed description of each aesthetic property, see docs for `element_text()`.

`nColumns`, `nRows` The dimensions of the grid to create - if both are NULL it will use the same logic as `facet_wrap()` to set the dimensions

`byRow` Analogous to `byrow` in `matrix()`. If FALSE the plots will be filled in in column-major order.

`widths`, `heights` The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid.

`guides` A string specifying how guides should be treated in the layout. 'collect' will collect guides below to the given nesting level, removing duplicates. 'keep' will stop collection at this level and let guides be placed alongside their plot. auto will allow guides to be collected if a upper level tries, but place them alongside the plot if not. If you modify default guide "position" with `theme(legend.position=...)` while also collecting guides you must apply that change to the overall patchwork.

`design` Specification of the location of areas in the layout. Can either be specified as a text string or by concatenating calls to `patchwork::area()` together. See the examples in `wrap_plots()` for further information on use.

Active bindings

`title`, `subtitle`, `caption` Text strings to use for the various plot annotations, where plot refers to the grid of plots as a whole.

`titleColor`, `titleSize`, `titleFontFace`, `titleFontFamily`, `titleHorizontalJustification`, `titleVerticalJustification` Aesthetic properties for the plot title.

`subtitleColor`, `subtitleSize`, `subtitleFontFace`, `subtitleFontFamily`, `subtitleHorizontalJustification`, `subtitleVerticalJustification` Aesthetic properties for the plot subtitle.

`captionColor`, `captionSize`, `captionFontFace`, `captionFontFamily`, `captionHorizontalJustification`, `captionVerticalJustification` Aesthetic properties for the plot caption.

`tagPrefix`, `tagSuffix` Strings that should appear before or after the tag.

`tagColor`, `tagSize`, `tagFontFamily`, `tagFontFace`, `tagHorizontalJustification`, `tagVerticalJustification`, `tagAngle` Aesthetic properties of individual plot tag text. For more detailed description of each aesthetic property, see docs for `element_text()`.

nColumns, nRows The dimensions of the grid to create - if both are NULL it will use the same logic as `facet_wrap()` to set the dimensions

widths, heights The relative widths and heights of each column and row in the grid. Will get repeated to match the dimensions of the grid.

Methods

Public methods:

- `PlotGridConfiguration$new()`
- `PlotGridConfiguration$addPlots()`
- `PlotGridConfiguration$print()`
- `PlotGridConfiguration$clone()`

Method `new()`: Create an instance of `PlotGridConfiguration` class.

Usage:

```
PlotGridConfiguration$new(plotList = NULL)
```

Arguments:

`plotList` A list containing ggplot objects.

Returns: A `PlotGridConfiguration` object.

Method `addPlots()`: Add a plot object.

Usage:

```
PlotGridConfiguration$addPlots(plots = NULL)
```

Arguments:

`plots` A single or a list containing ggplot object(s).

Returns: `PlotGridConfiguration` object with `$plotList` field updated to store entered plots.

Examples:

```
library(ggplot2)
```

```
myPlotGrid <- PlotGridConfiguration$new()
```

```
# You can add a single ggplot object
```

```
p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
```

```
myPlotGrid$addPlots(p)
```

```
# Or you can also pass a list
```

```
myPlotGrid$addPlots(list("p1" = ggplot(), "p2" = ggplot()))
```

```
# Since we added three plots, the `plotList` field should
```

```
# now be a list of length `3`
```

```
length(myPlotGrid$plotList)
```

Method `print()`: Print the object to the console.

Usage:

```
PlotGridConfiguration$print()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

Examples

```
library(tlf)

# create a list of plots
ls_plots <- list(
  # first plot
  plotBoxWhisker(mtcars,
    dataMapping = BoxWhiskerDataMapping$new(x = "am", y = "wt"), outliers = FALSE
  ),
  # second plot
  plotBoxWhisker(ToothGrowth,
    dataMapping = BoxWhiskerDataMapping$new(x = "supp", y = "len")
  )
)

plotGridObj <- PlotGridConfiguration$new(ls_plots)

# specify further customizations for the plot grid
plotGridObj$title <- "my combined plot"
plotGridObj$subtitle <- "something clever"
plotGridObj$caption <- "my sources"
plotGridObj$nColumns <- 2L
plotGridObj$tagLevels <- "A"
plotGridObj$tagPrefix <- "Plot ("
plotGridObj$tagSuffix <- ")"
plotGridObj$tagColor <- "blue"
plotGridObj$tagSize <- 15
plotGridObj$tagAngle <- 45
plotGridObj$tagPosition <- TagPositions$top
plotGridObj$titleHorizontalJustification <- HorizontalJustification$middle
plotGridObj$subtitleHorizontalJustification <- HorizontalJustification$middle
```

```

# print the object to see its properties
plotGridObj

## -----
## Method `PlotGridConfiguration$addPlots`
## -----

library(ggplot2)

myPlotGrid <- PlotGridConfiguration$new()

# You can add a single ggplot object
p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
myPlotGrid$addPlots(p)

# Or you can also pass a list
myPlotGrid$addPlots(list("p1" = ggplot(), "p2" = ggplot()))

# Since we added three plots, the `plotList` field should
# now be a list of length `3`
length(myPlotGrid$plotList)

```

plotHistogram

plotHistogram

Description

Producing Histograms

Usage

```

plotHistogram(
  data = NULL,
  metaData = NULL,
  x = NULL,
  dataMapping = NULL,
  frequency = NULL,
  bins = NULL,
  binwidth = NULL,
  stack = NULL,
  distribution = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)

```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
x	Numeric values to plot along the x axis. Only used instead of data if data is NULL.
dataMapping	A HistogramDataMapping object mapping x and aesthetic groups to their variable names of data.
frequency	logical defining if histogram displays a frequency in y axis
bins	Number or edges of bins. If bins is provided as a single numeric values, bin corresponds to number of bins. The bin edges are then equally spaced within the range of data. If bins is provided as an array of numeric values, bin corresponds to their edges. Default value, bins=NULL, uses the value defined by dataMapping
binwidth	Numerical value of defining the width of each bin. If defined, binwidth can overwrite bins if bins was not provided or simply provided as a single value. Default value, binwidth=NULL, uses the value defined by dataMapping
stack	Logical defining for multiple histograms if their bars are stacked Default value, stack=NULL, uses the value defined by dataMapping
distribution	Name of distribution to fit to the data. Only 2 distributions are currently available: "normal" and "logNormal" Use distribution="none" to prevent fit of distribution Default value, distribution=NULL, uses the value defined by dataMapping
plotConfiguration	An optional HistogramPlotConfiguration object defining labels, grid, background and watermark.
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/histogram.html>

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce histogram of normally distributed data
plotHistogram(x = rnorm(100))

# Produce histogram of normally distributed data normalized in y axis
plotHistogram(x = rnorm(100), frequency = TRUE)

# Produce histogram of normally distributed data with many bins
plotHistogram(x = rlnorm(100), bins = 21)

# Produce histogram of fitted normally distributed data
plotHistogram(x = rlnorm(100), distribution = "normal")

# Produce histogram of fitted normally distributed data
plotHistogram(x = rlnorm(100), distribution = "normal", frequency = TRUE, stack = TRUE)
```

```
plotObservedTimeProfile
      plotObservedTimeProfile
```

Description

Producing Time Profile plots for observed data

Usage

```
plotObservedTimeProfile(
  data,
  metaData = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	An ObservedDataMapping object mapping x, y, ymin, ymax and aesthetic groups to their variable names of observedData.
<code>plotConfiguration</code>	An optional TimeProfilePlotConfiguration object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce a Time profile plot with observed data
obsData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))
plotObservedTimeProfile(
  data = obsData,
  dataMapping = ObservedDataMapping$new(x = "x", y = "y")
)
```

plotObsVsPred

plotObsVsPred

Description

Producing observed vs predicted plots

Usage

```
plotObsVsPred(
  data,
  metaData = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  foldDistance = NULL,
  smoother = NULL,
  plotObject = NULL
)
```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
dataMapping	A ObsVsPredDataMapping object mapping x, y and aesthetic groups to their variable names of data.
plotConfiguration	An optional ObsVsPredConfiguration object defining labels, grid, background and watermark.

foldDistance	Numeric values of fold distance lines to display in log plots. This argument is internally translated into lines field of dataMapping. Caution: this argument is meant for log scaled plots and since fold distance is a ratio it is expected positive. In particular, line of identity corresponds to a foldDistance of 1.
smoother	Optional name of smoother function: • "loess" for loess regression • "lm" for linear regression
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce Obs vs Pred plot
obsVsPredData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

plotObsVsPred(data = obsVsPredData, dataMapping = ObsVsPredDataMapping$new(x = "x", y = "y"))

# Produce Obs vs Pred plot with linear regression
plotObsVsPred(
  data = obsVsPredData,
  dataMapping = ObsVsPredDataMapping$new(x = "x", y = "y"),
  smoother = "lm"
)

# Produce Obs vs Pred plot with user-defined fold distance lines
plotObsVsPred(
  data = obsVsPredData,
  dataMapping = ObsVsPredDataMapping$new(x = "x", y = "y"),
  plotConfiguration = ObsVsPredPlotConfiguration$new(
    xScale = Scaling$log, xAxisLimits = c(0.05, 50),
    yScale = Scaling$log, yAxisLimits = c(0.05, 50)
  ),
  foldDistance = c(1, 10)
)
```

`plotPieChart`*plotPieChart*

Description

Producing a Pie Chart

Usage

```
plotPieChart(  
  data = NULL,  
  metaData = NULL,  
  dataMapping = NULL,  
  colorPalette = NULL,  
  start = NULL,  
  clockwiseDirection = NULL,  
  plotConfiguration = NULL,  
  plotObject = NULL  
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	A PieChartDataMapping object mapping x and fill aesthetic groups to their variable names of data. Values mapped to y variable will be displayed as text within the pie chart
<code>colorPalette</code>	color palette property from ggplot2
<code>start</code>	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction
<code>clockwiseDirection</code>	logical defining if values are displayed in clockwise order
<code>plotConfiguration</code>	An optional PieChartPlotConfiguration object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: `plotBoxWhisker()`, `plotCumulativeTimeProfile()`, `plotDDIRatio()`, `plotGrid()`, `plotHistogram()`, `plotObsVsPred()`, `plotObservedTimeProfile()`, `plotPKRatio()`, `plotQQ()`, `plotResVsPred()`, `plotResVsTime()`, `plotSimulatedTimeProfile()`, `plotTimeProfile()`, `plotTornado()`

Examples

```
# Data for the pie chart
values <- runif(5)
data <- data.frame(
  values = values,
  text = paste0(round(100 * values / sum(values)), "%"),
  legend = letters[1:5]
)

# Plot pie chart with its legend
plotPieChart(
  data = data,
  dataMapping = PieChartDataMapping$new(x = "values", fill = "legend")
)

# Plot pie chart with text within pie
plotPieChart(
  data = data,
  dataMapping = PieChartDataMapping$new(x = "values", y = "text", fill = "legend")
)

# Reverse direction of pie chart
plotPieChart(
  data = data,
  dataMapping = PieChartDataMapping$new(x = "values", y = "text", fill = "legend"),
  clockwiseDirection = FALSE
)

# Start first slice of pie at 90 degrees
plotPieChart(
  data = data,
  dataMapping = PieChartDataMapping$new(x = "values", y = "text", fill = "legend"),
  start = pi / 2
)

# Leverages ggplot color palettes
plotPieChart(
  data = data,
  dataMapping = PieChartDataMapping$new(x = "values", y = "text", fill = "legend"),
  colorPalette = ColorPalettes$Set1
)
```

plotPKRatio*plotPKRatio*

Description

Producing PK Ratio plots

Usage

```
plotPKRatio(  
  data,  
  metaData = NULL,  
  dataMapping = NULL,  
  plotConfiguration = NULL,  
  foldDistance = NULL,  
  plotObject = NULL  
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	A PKRatioDataMapping object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional PKRatioPlotConfiguration object defining labels, grid, background and watermark.
<code>foldDistance</code>	Numeric values of fold distance lines to display in log plots. This argument is internally translated into lines field of dataMapping. Caution: this argument is meant for log scaled plots and since fold distance is a ratio it is expected positive. In particular, line of identity corresponds to a foldDistance of 1.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

References

For examples, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/pk-ratio-vignette.html>

See Also

Other molecule plots: `plotBoxWhisker()`, `plotCumulativeTimeProfile()`, `plotDDIRatio()`, `plotGrid()`, `plotHistogram()`, `plotObsVsPred()`, `plotObservedTimeProfile()`, `plotPieChart()`, `plotQQ()`, `plotResVsPred()`, `plotResVsTime()`, `plotSimulatedTimeProfile()`, `plotTimeProfile()`, `plotTornado()`

Examples

```
# Produce PK Ratio plot
pkData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

plotPKRatio(data = pkData, dataMapping = PKRatioDataMapping$new(x = "x", y = "y"))

# Produce PK Ratio plot with user-defined horizontal lines
plotPKRatio(
  data = pkData,
  dataMapping = PKRatioDataMapping$new(x = "x", y = "y"),
  foldDistance = c(1, 10)
)
```

plotQQ

plotQQ

Description

Producing Histograms

Usage

```
plotQQ(
  data = NULL,
  metaData = NULL,
  y = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>y</code>	Numeric values to plot along the y axis. Only used instead of data if data is NULL.

dataMapping	A QQDataMapping object mapping y and aesthetic groups to their variable names of data.
plotConfiguration	An optional QQPlotConfiguration object defining labels, grid, background and watermark.
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce QQ plot of normally distributed data
plotQQ(y = rnorm(100))

# Produce QQ plot of normally distributed data split by group
qqData <- data.frame(
  residuals = c(rnorm(100), rnorm(100)),
  groups = c(rep("Group A", 100), rep("Group B", 100))
)
plotQQ(
  data = qqData,
  dataMapping = QQDataMapping$new(y = "residuals", group = "groups")
)
```

plotResVsPred

plotResVsPred

Description

Producing residuals vs predicted plots

Usage

```
plotResVsPred(
  data,
  metaData = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
```

```

    smoother = NULL,
    plotObject = NULL
  )

```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	A ResVsPredDataMapping object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional ResVsPredConfiguration object defining labels, grid, background and watermark.
<code>smoother</code>	Optional name of smoother function: • "loess" for loess regression • "lm" for linear regression
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```

# Produce Obs vs Pred plot
resVsPredData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

plotResVsPred(data = resVsPredData, dataMapping = ResVsPredDataMapping$new(x = "x", y = "y"))

# Produce Res vs Pred plot with linear regression
plotResVsPred(
  data = resVsPredData,
  dataMapping = ResVsPredDataMapping$new(x = "x", y = "y"),
  smoother = "lm"
)

```

`plotResVsTime`*plotResVsTime*

Description

Producing residuals vs time plots

Usage

```
plotResVsTime(  
  data,  
  metaData = NULL,  
  dataMapping = NULL,  
  plotConfiguration = NULL,  
  smoother = NULL,  
  plotObject = NULL  
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	A ResVsTimeDataMapping object mapping x, y and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional ResVsTimeConfiguration object defining labels, grid, background and watermark.
<code>smoother</code>	Optional name of smoother function: • "loess" for loess regression • "lm" for linear regression
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce Obs vs Pred plot
resVsTimeData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))

plotResVsTime(data = resVsTimeData, dataMapping = ResVsTimeDataMapping$new(x = "x", y = "y"))

# Produce Res vs Time plot with linear regression
plotResVsTime(
  data = resVsTimeData,
  dataMapping = ResVsTimeDataMapping$new(x = "x", y = "y"),
  smoother = "lm"
)
```

```
plotSimulatedTimeProfile
      plotSimulatedTimeProfile
```

Description

Producing Time Profile plots

Usage

```
plotSimulatedTimeProfile(
  data = NULL,
  metaData = NULL,
  dataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

<code>data</code>	A data.frame to use for plot.
<code>metaData</code>	A named list of information about data such as the dimension and unit of its variables.
<code>dataMapping</code>	A TimeProfileDataMapping object mapping x, y, ymin, ymax and aesthetic groups to their variable names of data.
<code>plotConfiguration</code>	An optional TimeProfilePlotConfiguration object defining labels, grid, background and watermark.
<code>plotObject</code>	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce a Time profile plot with simulated data
simTime <- seq(1, 10, 0.1)
simData <- data.frame(
  x = simTime,
  y = 10 * exp(-simTime),
  ymin = 8 * exp(-simTime),
  ymax = 12 * exp(-simTime)
)

plotSimulatedTimeProfile(
  data = simData,
  dataMapping = TimeProfileDataMapping$new(x = "x", y = "y", ymin = "ymin", ymax = "ymax")
)
```

plotTimeProfile	<i>plotTimeProfile</i>
-----------------	------------------------

Description

Producing Time Profile plots

Usage

```
plotTimeProfile(
  data = NULL,
  metaData = NULL,
  dataMapping = NULL,
  observedData = NULL,
  observedDataMapping = NULL,
  plotConfiguration = NULL,
  plotObject = NULL
)
```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
dataMapping	A TimeProfileDataMapping object mapping x, y, ymin, ymax and aesthetic groups to their variable names of data.

observedData	A data.frame to use for plot. Unlike data, meant for simulated data, plotted as lines and ribbons; observedData is plotted as scatter points and errorbars.
observedDataMapping	An ObservedDataMapping object mapping x, y, ymin, ymax and aesthetic groups to their variable names of observedData.
plotConfiguration	An optional TimeProfilePlotConfiguration object defining labels, grid, background and watermark.
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTornado\(\)](#)

Examples

```
# Produce a Time profile plot with observed and simulated data
obsData <- data.frame(x = c(1, 2, 1, 2, 3), y = c(5, 0.2, 2, 3, 4))
simTime <- seq(1, 10, 0.1)
simData <- data.frame(
  x = simTime,
  y = 10 * exp(-simTime),
  ymin = 8 * exp(-simTime),
  ymax = 12 * exp(-simTime)
)

plotTimeProfile(
  data = simData,
  observedData = obsData,
  dataMapping = TimeProfileDataMapping$new(x = "x", y = "y", ymin = "ymin", ymax = "ymax"),
  observedDataMapping = ObservedDataMapping$new(x = "x", y = "y")
)
```

plotTornado

plotTornado

Description

Producing tornado plots

Usage

```
plotTornado(  
  data = NULL,  
  metaData = NULL,  
  x = NULL,  
  y = NULL,  
  sorted = NULL,  
  colorPalette = NULL,  
  bar = TRUE,  
  dataMapping = NULL,  
  plotConfiguration = NULL,  
  plotObject = NULL  
)
```

Arguments

data	A data.frame to use for plot.
metaData	A named list of information about data such as the dimension and unit of its variables.
x	Numeric values to plot along the x axis. Only used instead of data if data is NULL.
y	Character values to plot along the y axis. Only used instead of data if data is NULL.
sorted	Optional logical value defining if y values are sorted by absolute values of x.
colorPalette	Optional character values defining a ggplot2 colorPalette (e.g. "Spectral")
bar	Optional logical value setting tornado plot as bar plot instead of scatter plot.
dataMapping	A TornadoDataMapping object mapping x, y and aesthetic groups to their variable names of data.
plotConfiguration	An optional TornadoPlotConfiguration object defining labels, grid, background and watermark.
plotObject	An optional ggplot object on which to add the plot layer

Value

A ggplot object

See Also

Other molecule plots: [plotBoxWhisker\(\)](#), [plotCumulativeTimeProfile\(\)](#), [plotDDIRatio\(\)](#), [plotGrid\(\)](#), [plotHistogram\(\)](#), [plotObsVsPred\(\)](#), [plotObservedTimeProfile\(\)](#), [plotPKRatio\(\)](#), [plotPieChart\(\)](#), [plotQQ\(\)](#), [plotResVsPred\(\)](#), [plotResVsTime\(\)](#), [plotSimulatedTimeProfile\(\)](#), [plotTimeProfile\(\)](#)

Examples

```
# Produce a tornado plot
plotTornado(x = c(2, -1, 3), y = c("A", "B", "C"))

# Produce a tornado plot as scatter plot
plotTornado(x = c(2, -1, 3), y = c("A", "B", "C"), bar = FALSE)

# Produce a tornado plot as is (no sorting)
plotTornado(x = c(2, -1, 3), y = c("A", "B", "C"), sorted = FALSE)
```

QQDataMapping

QQDataMapping

Description

R6 class for mapping x, y and GroupMapping variables to data

Super classes

tlf::XYDataMapping -> tlf::XYGDataMapping -> QQDataMapping

Methods**Public methods:**

- [QQDataMapping\\$checkMapData\(\)](#)
- [QQDataMapping\\$clone\(\)](#)

Method checkMapData(): Check that data variables include map variables

Usage:

```
QQDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check

metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

QQPlotConfiguration	<i>QQPlotConfiguration</i>
---------------------	----------------------------

Description

R6 class defining the configuration of a ggplot object for Quantile-Quantile plots

Super class

`tlf::PlotConfiguration -> QQPlotConfiguration`

Methods

Public methods:

- [QQPlotConfiguration\\$new\(\)](#)
- [QQPlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new QQPlotConfiguration object

Usage:

`QQPlotConfiguration$new(xlabel = "Standard Normal Quantiles", ...)`

Arguments:

`xlabel` QQ-plot default display is "Standard Normal Quantiles"

`...` parameters inherited from PlotConfiguration

Returns: A new QQPlotConfiguration object

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

Usage:

`QQPlotConfiguration$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

RangeDataMapping	<i>RangeDataMapping</i>
------------------	-------------------------

Description

R6 class for mapping x, ymin and ymax variable to data

Super classes

tlf::XYDataMapping -> tlf::XYGDataMapping -> RangeDataMapping

Public fields

ymin Name of ymin variable to map

ymax Name of ymax variable to map

Methods**Public methods:**

- [RangeDataMapping\\$new\(\)](#)
- [RangeDataMapping\\$checkMapData\(\)](#)
- [RangeDataMapping\\$clone\(\)](#)

Method new(): Create a new RangeDataMapping object

Usage:

```
RangeDataMapping$new(
  x = NULL,
  ymin = NULL,
  ymax = NULL,
  groupMapping = NULL,
  color = NULL,
  fill = NULL,
  linetype = NULL,
  shape = NULL,
  size = NULL,
  group = NULL,
  data = NULL
)
```

Arguments:

x Name of x variable to map

ymin Name of ymin variable to map

ymax Name of ymax variable to map

groupMapping R6 class GroupMapping object

color R6 class Grouping object or its input

fill R6 class Grouping object or its input

linetype R6 class Grouping object or its input
 shape R6 class Grouping object or its input
 size R6 class Grouping object or its input
 group R6 class Grouping object or its input
 data data.frame to map used by .smartMapping
Returns: A new RangeDataMapping object

Method checkMapData(): Check that data variables include map variables

Usage:

```
RangeDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check
 metaData list containing information on data

Returns: A data.frame with map and legendLabels variables. Dummy variable legendLabels is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

```
resetTLFSettingsToDefault
```

```
resetTLFSettingsToDefault
```

Description

Reset the global settings stored in tlfEnv to default values defined by the package.

Usage

```
resetTLFSettingsToDefault()
```

ResVsPredDataMapping *ResVsPredDataMapping*

Description

Class for mapping variables in residuals vs predictions/time plot

Super classes

`tlf::XYDataMapping -> tlf::XYGDataMapping -> tlf::ObsVsPredDataMapping -> ResVsPredDataMapping`

Methods

Public methods:

- [ResVsPredDataMapping\\$clone\(\)](#)

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

Usage:

`ResVsPredDataMapping$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

ResVsPredPlotConfiguration

ResVsPredPlotConfiguration

Description

R6 class defining the configuration of a ggplot object for Res vs Pred/Time plots

Super class

`tlf::PlotConfiguration -> ResVsPredPlotConfiguration`

Public fields

`defaultSymmetricAxes` Default option setting symmetric xAxis and/or yAxis limits when creating a ResVsPredPlotConfiguration object

Methods**Public methods:**

- [ResVsPredPlotConfiguration\\$clone\(\)](#)

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

ResVsTimeDataMapping *ResVsTimeDataMapping*

Description

Class for mapping variables in residuals vs predictions/time plot

Super classes

```
tlf::XYDataMapping -> tlf::XYGDataMapping -> tlf::ObsVsPredDataMapping -> tlf::ResVsPredDataMapping
-> ResVsTimeDataMapping
```

Methods**Public methods:**

- [ResVsTimeDataMapping\\$clone\(\)](#)

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

ResVsTimePlotConfiguration

ResVsTimePlotConfiguration

Description

R6 class defining the configuration of a ggplot object for Res vs Pred/Time plots

Super classes

`tlf::PlotConfiguration -> tlf::ResVsPredPlotConfiguration -> ResVsTimePlotConfiguration`

Methods**Public methods:**

- [ResVsTimePlotConfiguration\\$clone\(\)](#)

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

Usage:

`ResVsTimePlotConfiguration$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

`runPlotMaker`*runPlotMaker*

Description

Run shiny app that allows to easily make plots from user interface.

Usage

```
runPlotMaker()
```

References

For tutorial, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/plot-maker.html>

See Also

Other shiny apps: [runThemeMaker\(\)](#)

`runThemeMaker`*runThemeMaker*

Description

Run shiny app that allows easy setting of Theme objects. Theme objects drive default properties of plots

Usage

```
runThemeMaker()
```

References

For tutorial, see: <https://www.open-systems-pharmacology.org/TLF-Library/articles/theme-maker.html>

See Also

Other shiny apps: [runPlotMaker\(\)](#)

saveThemeToJson	<i>saveThemeToJson</i>
-----------------	------------------------

Description

Save theme object to a json file.

Usage

```
saveThemeToJson(jsonFile, theme = NULL)
```

Arguments

jsonFile	path of json file
theme	Theme object path of json file

saveTLFSettings	<i>saveTLFSettings</i> Save the current TLF global settings in a .RData file
-----------------	--

Description

saveTLFSettings Save the current TLF global settings in a .RData file

Usage

```
saveTLFSettings(file)
```

Arguments

file	.RData file containing the settings
------	-------------------------------------

Scaling	<i>Scaling</i>
---------	----------------

Description

Helper enum of predefined transformations of axes Note that the transformations will be translated internally into ggplot2 transformations. ggplot2 includes more transformations than what is available in this enum.

Usage

Scaling

Format

An object of class list of length 9.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Examples

```
# Continuous linear/identity scale
Scaling$identity
Scaling$lin

# Continuous log10 scale
Scaling$log

# Continuous natural logarithm (ln) scale (base is *e*)
Scaling$ln

# Discrete scale for categorical data such as boxplot and tornado plot data
Scaling$discrete

# Reverse continuous linear scale to switch end and beginning of linear scale
Scaling$reverse

# Continuous square root scale
Scaling$sqrt

# Time scale for POSIXlt or POSIXct data
Scaling$time

# Date scale for POSIXlt or POSIXct data
Scaling$date
```

setBackground

setBackground

Description

Set background properties of a ggplot object

Usage

```
setBackground(  
  plotObject,  
  fill = NULL,  
  color = NULL,  
  linetype = NULL,  
  size = NULL  
)
```

Arguments

plotObject	A ggplot object
fill	Optional character values defining the color of the background. See <code>grDevices::colors()</code> to get names of colors
color	Optional character values defining the color of the background frame. See <code>grDevices::colors()</code> to get names of colors
linetype	Optional character values defining the linetype of the background frame. See <code>enum Linetypes</code> to get names of linetype.
size	Optional numeric values defining the size of the background frame.

Value

A ggplot object

Examples

```
# Set background of a scatter plot  
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))  
  
setBackground(p, fill = "yellowgreen", color = "red", linetype = "dotted")
```

setBackgroundPanelArea

setBackgroundPanelArea

Description

Set background panel area properties of a ggplot object

Usage

```
setBackgroundPanelArea(
  plotObject,
  fill = NULL,
  color = NULL,
  linetype = NULL,
  size = NULL
)
```

Arguments

plotObject	A ggplot object
fill	Optional character values defining the color of the background. See <code>grDevices::colors()</code> to get names of colors
color	Optional character values defining the color of the background frame. See <code>grDevices::colors()</code> to get names of colors
linetype	Optional character values defining the linetype of the background frame. See <code>enum Linetypes</code> to get names of linetype.
size	Optional numeric values defining the size of the background frame.

Value

A ggplot object

Examples

```
# Set background of a scatter plot
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setBackgroundPanelArea(p, fill = "yellowgreen", color = "red", linetype = "dotted")
```

setBackgroundPlotArea *setBackgroundPlotArea*

Description

Set background plot area properties of a ggplot object

Usage

```
setBackgroundPlotArea(
  plotObject,
  fill = NULL,
  color = NULL,
  linetype = NULL,
  size = NULL
)
```

Arguments

plotObject	A ggplot object
fill	Optional character values defining the color of the background. See grDevices::colors() to get names of colors
color	Optional character values defining the color of the background frame. See grDevices::colors() to get names of colors
linetype	Optional character values defining the linetype of the background frame. See enum Linetypes to get names of linetype.
size	Optional numeric values defining the size of the background frame.

Value

A ggplot object

Examples

```
# Set background of a scatter plot
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setBackgroundPlotArea(p, fill = "yellowgreen", color = "red", linetype = "dotted")
```

setCaptionColor	<i>setCaptionColor</i>
-----------------	------------------------

Description

Set the colors of the data in plot and legend caption

Usage

```
setCaptionColor(plotObject, color, name = NULL)
```

Arguments

plotObject	ggplot graphical object
color	colors of the data in plot and legend caption
name	reference names in caption\$name identifying the legend caption lines

Value

A ggplot graphical object

setCaptionFill	<i>setCaptionFill</i>
----------------	-----------------------

Description

Set the fills of the data in plot and legend caption

Usage

```
setCaptionFill(plotObject, fill, name = NULL)
```

Arguments

- | | |
|------------|---|
| plotObject | ggplot graphical object |
| fill | fills of the data in plot and legend caption |
| name | reference names in caption\$name identifying the legend caption lines |

Value

A ggplot graphical object

setCaptionLabels	<i>setCaptionLabel</i>
------------------	------------------------

Description

Set the legend caption labels

Usage

```
setCaptionLabels(plotObject, label, name = NULL)
```

Arguments

- | | |
|------------|---|
| plotObject | ggplot graphical object |
| label | new labels of the legend caption |
| name | reference names in caption\$name identifying the legend caption lines |

Value

A ggplot graphical object

setCaptionLinetype	<i>setCaptionLinetype</i>
--------------------	---------------------------

Description

Set the linetypes of the data in plot and legend caption

Usage

```
setCaptionLinetype(plotObject, linetype, name = NULL)
```

Arguments

- | | |
|------------|---|
| plotObject | ggplot graphical object |
| linetype | linetypes of the data in plot and legend caption |
| name | reference names in caption\$name identifying the legend caption lines |

Value

A ggplot graphical object

setCaptionOrder	<i>setCaptionOrder</i>
-----------------	------------------------

Description

Set the order of the legend caption labels

Usage

```
setCaptionOrder(plotObject, order, name = NULL)
```

Arguments

- | | |
|------------|---|
| plotObject | ggplot graphical object |
| order | numeric order of the legend caption labels |
| name | reference names in caption\$name identifying the legend caption lines |

Value

A ggplot graphical object

setCaptionShape	<i>setCaptionShape</i>
-----------------	------------------------

Description

Set the shapes of the data in plot and legend caption

Usage

```
setCaptionShape(plotObject, shape, name = NULL)
```

Arguments

plotObject	ggplot graphical object
shape	shapes of the data in plot and legend caption
name	reference names in caption\$name identifying the legend caption lines

Value

A ggplot graphical object

setCaptionSize	<i>setCaptionSize</i>
----------------	-----------------------

Description

Set the sizes of the data in plot and legend caption

Usage

```
setCaptionSize(plotObject, size, name = NULL)
```

Arguments

plotObject	ggplot graphical object
size	sizes of the data in plot and legend caption
name	reference names in caption\$name identifying the legend caption lines

Value

A ggplot graphical object

`setCaptionVisibility` *setCaptionVisibility*

Description

Set the visibility of the legend caption labels

Usage

```
setCaptionVisibility(plotObject, visibility, name = NULL)
```

Arguments

<code>plotObject</code>	ggplot graphical object
<code>visibility</code>	logical visibility of the legend caption labels
<code>name</code>	reference names in <code>caption\$name</code> identifying the legend caption lines

Value

A ggplot graphical object

`setDefaultAggregationBins`
setDefaultAggregationBins

Description

Set default aggregation bins of tlf environment

Usage

```
setDefaultAggregationBins(bins = NULL)
```

Arguments

<code>bins</code>	Number of bins if value, edges if vector or binning function if function
-------------------	--

Examples

```
# Set default number of bins
plotHistogram(x = rnorm(1000))

setDefaultAggregationBins(21)
plotHistogram(x = rnorm(1000))
```

```
setDefaultAggregationFunctions  
    setDefaultAggregationFunctions
```

Description

Set default aggregation functions of tlf environment

Usage

```
setDefaultAggregationFunctions(y = NULL, ymin = NULL, ymax = NULL)
```

Arguments

y	function or its name as median aggregation
ymin	function or its name as min aggregation
ymax	function or its name as max aggregation

```
setDefaultAggregationLabels  
    setDefaultAggregationLabels
```

Description

Set default aggregation labels of tlf environment

Usage

```
setDefaultAggregationLabels(y = NULL, range = NULL)
```

Arguments

y	label for median aggregation
range	label for range aggregation

`setDefaultAlphaRatio` *setDefaultAlphaRatio*

Description

Set the default alpha (transparency) difference ratio between points above and below lloq

Usage

```
setDefaultAlphaRatio(alphaRatio)
```

Arguments

`alphaRatio` alpha ratio to set as default. Must be between 0 and 1.

`setDefaultErrorbarCapSize`
setDefaultErrorbarCapSize

Description

Set default cap size of error bars

Usage

```
setDefaultErrorbarCapSize(size)
```

Arguments

`size` A numeric defining the size of the error bar caps in pts

`setDefaultExportParameters`
setDefaultExportParameters

Description

Set default tlf properties for exporting/saving plots

Usage

```
setDefaultExportParameters(  
    format = NULL,  
    width = NULL,  
    height = NULL,  
    units = NULL,  
    dpi = NULL,  
    name = NULL  
)
```

Arguments

format	file format of the exported plots
width	plot width in unit
height	plot height in unit
units	units of width and height
dpi	units of width and height
name	base file name of the exported plots

```
setDefaultLegendPosition  
    setDefaultLegendPosition
```

Description

Set default legend position of tlf environment

Usage

```
setDefaultLegendPosition(position)
```

Arguments

position	legend position. Use Enum LegendPositions to assess available legend positions
----------	--

setDefaultLegendTitle	<i>setDefaultLegendTitle</i>
-----------------------	------------------------------

Description

Set default legend title of tlf environment

Usage

setDefaultLegendTitle(title)

Arguments

title	Character or Label object
-------	---------------------------

setDefaultLLOQLinetype	<i>setDefaultLLOQLinetype</i>
------------------------	-------------------------------

Description

Set default cap linetype for lloq lines

Usage

setDefaultLLOQLinetype(linetype)

Arguments

linetype	linetype to set as default
----------	----------------------------

setDefaultLogTicks	<i>setDefaultLogTicks</i>
--------------------	---------------------------

Description

Set default values for log ticks

Usage

setDefaultLogTicks(ticks)

Arguments

ticks	numeric values where ticks are placed. Ensure that the values are positive (they are meant for log scale)
-------	---

setDefaultMaxCharacterWidth	
	<i>setDefaultMaxCharacterWidth</i>

Description

Set the maximum number of characters per row for axis ticks labels and legend. Long strings will be wrapped on spaces or non-word characters.

Usage

```
setDefaultMaxCharacterWidth(maxCharacterWidth)
```

Arguments

maxCharacterWidth	
	the maximum number of characters per row

setDefaultWatermark	<i>setDefaultWatermark</i>
---------------------	----------------------------

Description

Set default watermark value for current theme

Usage

```
setDefaultWatermark(watermark = NULL)
```

Arguments

watermark	A character value or Label object
-----------	-----------------------------------

Examples

```
# Set default watermark using a character
setDefaultWatermark("Confidential")
addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

# Set default watermark using a `Label` object
setDefaultWatermark(Label$new(text = "Confidential", color = "red", angle = 30))
addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))
```

`setGrid`*setGrid*

Description

Set x and y grid properties of a ggplot object

Usage

```
setGrid(plotObject, color = NULL, linetype = NULL, size = NULL)
```

Arguments

<code>plotObject</code>	A ggplot object
<code>color</code>	Optional character values defining the color of the grid. See <code>grDevices::colors()</code> to get names of colors
<code>linetype</code>	Optional character values defining the linetype of the grid. See <code>enum Linetypes</code> to get names of linetype.
<code>size</code>	Optional numeric values defining the size of the grid.

Value

A ggplot object

Examples

```
# Set grid of a scatter plot
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setGrid(p, color = "red", linetype = "dotted")
```

`setLegend`*setLegend*

Description

Set legend position, title, font and/or caption

Usage

```
setLegend(
  plotObject,
  position = NULL,
  title = NULL,
  font = NULL,
  caption = NULL
)
```

Arguments

plotObject	Graphical object created from ggplot
position	legend position. Use enum LegendPositions to access the list of legend positions.
title	character or Label object
font	Font object defining legend font
caption	data.frame containing the caption properties of the legend

Value

A ggplot object

setLegendCaption	<i>setLegendCaption</i>
------------------	-------------------------

Description

Set the legend caption

Usage

```
setLegendCaption(plotObject, caption = NULL)
```

Arguments

plotObject	ggplot graphical object
caption	data.frame containing the caption properties of the legend

Value

A ggplot graphical object

setLegendFont	<i>setLegendFont</i>
---------------	----------------------

Description

Set legend font properties

Usage

```
setLegendFont(  
  plotObject,  
  color = NULL,  
  size = NULL,  
  fontFamily = NULL,  
  fontFace = NULL,  
  angle = NULL,  
  align = NULL  
)
```

Arguments

plotObject	ggplot object
color	character defining the color of legend font
size	numeric defining the size of legend font
fontFamily	character defining the family of legend font
fontFace	character defining the legend font face as defined in helper enum FontFaces.
angle	numeric defining the angle of legend font
align	character defining the alignment of legend font as defined in helper enum Alignments.

Value

A ggplot object

setLegendPosition	<i>setLegendPosition</i>
-------------------	--------------------------

Description

Set the legend position

Usage

```
setLegendPosition(plotObject, position = NULL)
```

Arguments

- plotObject ggplot graphical object
- position legend position as defined in helper enum LegendPositions

Value

A ggplot graphical object

See Also

LegendPositions

setLegendTitle	<i>setLegendTitle</i>
----------------	-----------------------

Description

Set legend title

Usage

```
setLegendTitle(plotObject, title = NULL)
```

Arguments

- plotObject ggplot object
- title character or Label object

Value

A ggplot object

setPlotExport	<i>setPlotExport</i>
---------------	----------------------

Description

Set plot export properties

Usage

```
setPlotExport(  
  plotObject,  
  name = NULL,  
  format = NULL,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL  
)
```

Arguments

plotObject	Graphical object created from ggplot
name	character defining the name of the file to be saved (without extension)
format	character defining the format of the file to be saved.
width	numeric values defining the width in units of the plot dimensions after saving
height	numeric values defining the height in units of the plot dimensions after saving
units	character defining the unit of the saving dimension
dpi	numeric value defining plot resolution (dots per inch)

Value

ggplot object with updated saving properties

setPlotExportDimensions
setPlotExportDimensions

Description

Set plot export properties

Usage

```
setPlotExportDimensions(  
  plotObject,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL  
)
```

Arguments

plotObject	Graphical object created from ggplot
width	plot width in unit
height	plot height in unit
units	units of width and height
dpi	numeric value defining plot resolution (dots per inch)

Value

ggplot object with updated labels

setPlotExportFormat	<i>setPlotExportFormat</i>
---------------------	----------------------------

Description

Set plot export properties

Usage

```
setPlotExportFormat(plotObject, format = NULL)
```

Arguments

plotObject	Graphical object created from ggplot
format	file format of the exported plot

Value

ggplot object with updated labels

setPlotExportSize	<i>setPlotExportSize</i>
-------------------	--------------------------

Description

Set plot export properties

Usage

```
setPlotExportSize(  
  plotObject,  
  width = NULL,  
  height = NULL,  
  units = NULL,  
  dpi = NULL  
)
```

Arguments

plotObject	Graphical object created from ggplot
width	plot width in unit
height	plot height in unit
units	units of width and height
dpi	numeric value defining plot resolution (dots per inch)

Value

ggplot object with updated labels

setPlotLabels	<i>setPlotLabels</i>
---------------	----------------------

Description

Set labels properties on a ggplot object

Usage

```
setPlotLabels(  
  plotObject,  
  title = NULL,  
  subtitle = NULL,  
  xlabel = NULL,  
  ylabel = NULL,  
  caption = NULL  
)
```

Arguments

plotObject	A ggplot object
title	A character value or Label object
subtitle	A character value or Label object
xlabel	A character value or Label object
ylabel	A character value or Label object
caption	A character value or Label object

Value

A ggplot object

Examples

```
# Set labels of a scatter plot
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setPlotLabels(p, xlabel = "new x label", ylabel = "new y label")

# Set labels using Label object
setPlotLabels(p, ylabel = Label$new(text = "red y label", color = "red"))
```

setWatermark	<i>setWatermark</i>
--------------	---------------------

Description

Set the watermark of a ggplot object. Unlike addWatermark, the watermark layer is overridden by setWatermark.

Usage

```
setWatermark(
  plotObject,
  watermark = NULL,
  color = NULL,
  size = NULL,
  angle = NULL,
  alpha = NULL
)
```

Arguments

plotObject	A ggplot object
watermark	A character value or a Label object
color	Color of the watermark.
size	Size of the watermark.
angle	Angle of the watermark (in degree).
alpha	Numeric value between 0 and 1 corresponding to transparency of the watermark The closer to 0, the more transparent the watermark is. The closer to 1, the more opaque the watermark is.

Value

A ggplot object

Examples

```
# Add a watermark to an empty plot
p <- initializePlot()
setWatermark(p, "watermark")

# Watermark with font properties
watermarkLabel <- Label$new(text = "watermark", color = "blue")
setWatermark(p, watermarkLabel)

# Horizontal watermark
setWatermark(p, watermarkLabel, angle = 0)

# Watermark totally opaque
setWatermark(p, watermarkLabel, alpha = 1)
```

setXAxis	<i>setXAxis</i>
----------	-----------------

Description

Set X-axis properties of a ggplot object

Usage

```
setXAxis(
  plotObject,
  scale = NULL,
  valuesLimits = NULL,
  axisLimits = NULL,
  limits = lifecycle::deprecated(),
  ticks = NULL,
  ticklabels = NULL,
  minorTicks = NULL,
  font = NULL,
  expand = NULL
)
```

Arguments

plotObject	A ggplot object to set X-axis properties
scale	Scale of axis. Use enum Scaling to access names of scales.
valuesLimits	Optional numeric values of values limits
axisLimits	Optional numeric values of axis limits
limits	[Deprecated] . Replaced by axisLimits argument.
ticks	Optional values or function for axis ticks

ticklabels	Optional values or function for axis ticklabels
minorTicks	Optional values or function for axis minor ticks
font	A Font object defining font of ticklabels
expand	Logical defining if data is expanded until axis

Value

A ggplot object

Examples

```
myPlot <- addLine(x = c(1, 2, 3), y = c(10, 50, 100))

# Set x-axis in log scale
setXAxis(myPlot, scale = Scaling$log)

# Set x-axis ticklabels to Greek letters
setXAxis(myPlot, ticks = c(1, 2, 3), ticklabels = parse(text = c("alpha", "beta", "gamma")))

# Set x-axis limits
setXAxis(myPlot, axisLimits = c(1, 2.5))

# Set x-axis fonts
setXAxis(myPlot, font = Font$new(color = "blue", size = 14))
```

setXGrid	<i>setXGrid</i>
----------	-----------------

Description

Set x grid properties of a ggplot object

Usage

```
setXGrid(plotObject, color = NULL, linetype = NULL, size = NULL)
```

Arguments

plotObject	A ggplot object
color	Optional character values defining the color of the grid. See grDevices::colors() to get names of colors
linetype	Optional character values defining the linetype of the grid. See enum Linetypes to get names of linetype.
size	Optional numeric values defining the size of the grid.

Value

A ggplot object

Examples

```
# Set x grid of a scatter plot
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setXGrid(p, color = "red", linetype = "dotted")
```

setY2Axis	<i>setY2Axis</i>
-----------	------------------

Description

Set right Y-axis properties of a ggplot object

Usage

```
setY2Axis(
  plotObject,
  scale = NULL,
  valuesLimits = NULL,
  axisLimits = NULL,
  limits = lifecycle::deprecated(),
  ticks = NULL,
  ticklabels = NULL,
  minorTicks = NULL,
  font = NULL,
  expand = NULL
)
```

Arguments

plotObject	A ggplot object to set X-axis properties
scale	Scale of axis. Use enum Scaling to access names of scales.
valuesLimits	Optional numeric values of values limits
axisLimits	Optional numeric values of axis limits
limits	[Deprecated] . Replaced by axisLimits argument.
ticks	Optional values or function for axis ticks
ticklabels	Optional values or function for axis ticklabels
minorTicks	Optional values or function for axis minor ticks
font	A Font object defining font of ticklabels
expand	Logical defining if data is expanded until axis

Value

A ggplot object

`setYAxis`*setYAxis*

Description

Set Y-axis properties of a ggplot object

Usage

```
setYAxis(  
  plotObject,  
  scale = NULL,  
  valuesLimits = NULL,  
  axisLimits = NULL,  
  limits = lifecycle::deprecated(),  
  ticks = NULL,  
  ticklabels = NULL,  
  minorTicks = NULL,  
  font = NULL,  
  expand = NULL  
)
```

Arguments

<code>plotObject</code>	A ggplot object to set X-axis properties
<code>scale</code>	Scale of axis. Use enum <code>Scaling</code> to access names of scales.
<code>valuesLimits</code>	Optional numeric values of values limits
<code>axisLimits</code>	Optional numeric values of axis limits
<code>limits</code>	[Deprecated] . Replaced by <code>axisLimits</code> argument.
<code>ticks</code>	Optional values or function for axis ticks
<code>ticklabels</code>	Optional values or function for axis ticklabels
<code>minorTicks</code>	Optional values or function for axis minor ticks
<code>font</code>	A Font object defining font of ticklabels
<code>expand</code>	Logical defining if data is expanded until axis

Value

A ggplot object

Examples

```
myPlot <- addLine(x = c(1, 2, 3), y = c(10, 50, 100))  
  
# Set y-axis in log scale  
setYAxis(myPlot, scale = Scaling$log)
```

```
# Set y-axis ticklabels to Greek letters
setYAxis(myPlot, ticks = c(10, 50, 100), ticklabels = parse(text = c("alpha", "beta", "gamma")))

# Set y-axis limits
setYAxis(myPlot, axisLimits = c(10, 75))

# Set y-axis fonts
setYAxis(myPlot, font = Font$new(color = "blue", size = 14))
```

setYGrid	<i>setYGrid</i>
----------	-----------------

Description

Set y grid properties of a ggplot object

Usage

```
setYGrid(plotObject, color = NULL, linetype = NULL, size = NULL)
```

Arguments

plotObject	A ggplot object
color	Optional character values defining the color of the grid. See <code>grDevices::colors()</code> to get names of colors
linetype	Optional character values defining the linetype of the grid. See <code>enum Linetypes</code> to get names of linetype.
size	Optional numeric values defining the size of the grid.

Value

A ggplot object

Examples

```
# Set y grid of a scatter
p <- addScatter(x = c(1, 2, 1, 2, 3), y = c(5, 0, 2, 3, 4))

setYGrid(p, color = "red", linetype = "dotted")
```

*Shapes**Shapes*

Description

List of some ggplot2 shapes. The shapes from this enum/list are unicode characters corresponding to their appropriate shapes. Note that user-defined characters are also accepted by `geomTLPPoint()`

Usage

`Shapes`

Format

An object of class `list` of length 40.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Examples

```
# Use ggplot2 to plot and label shapes
shapesData <- data.frame(
  x = (seq_along(Shapes) - 1) %% 6,
  y = floor((seq_along(Shapes) - 1) / 6),
  shape = factor(names(Shapes), levels = names(Shapes))
)
ggplot2::ggplot(data = shapesData, ggplot2::aes(x, y)) +
  ggplot2::theme_void() +
  # Define size and color of shapes
  geomTLPPoint(ggplot2::aes(shape = shape), size = 8, color = "red") +
  # Add shape names from enum below the displayed shape
  ggplot2::geom_text(ggplot2::aes(label = shape), nudge_y = -0.3, size = 3) +
  # Use scale to display the actual shape
  ggplot2::scale_shape_manual(values = as.character(unlist(Shapes))) +
  # Remove the legend as the shape name is labelled below the shape
  ggplot2::guides(shape = "none")

# Perform a scatter plot with blue pentagons as shape
addScatter(
  x = 1:10,
  y = rlnorm(10),
  shape = Shapes$pentagon,
  color = "blue",
```

```
    size = 3
  )
```

TagPositions	<i>TagPositions</i>
--------------	---------------------

Description

List of all available tag positions in a plot grid.

Usage

TagPositions

Format

An object of class list of length 8.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

Theme	<i>Theme</i>
-------	--------------

Description

R6 class defining theme properties

Active bindings

fonts ThemeFont object
background ThemeBackground object
aestheticMaps ThemeAestheticMaps object
plotConfigurations ThemePlotConfiguration object

Methods

Public methods:

- [Theme\\$new\(\)](#)
- [Theme\\$save\(\)](#)
- [Theme\\$clone\(\)](#)

Method new(): Create a new Theme object

Usage:

```
Theme$new(  
  fonts = NULL,  
  background = NULL,  
  aestheticMaps = NULL,  
  plotConfigurations = NULL  
)
```

Arguments:

fonts ThemeFont object

background ThemeBackground object

aestheticMaps ThemeAestheticMaps object

plotConfigurations ThemePlotConfiguration object

Returns: A new Theme object

Method save(): Save Theme as a json file

Usage:

```
Theme$save(jsonFile)
```

Arguments:

jsonFile name of json file

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

ThemeAestheticMaps	<i>ThemeAestheticMaps</i>
--------------------	---------------------------

Description

R6 class defining theme aesthetic maps

Public fields

color color map as character or numeric vector
 fill fill map as character or numeric vector
 size size map as numeric vector
 shape shape map as numeric vector
 linetype linetype as character vector
 alpha map as numeric vector

Methods**Public methods:**

- [ThemeAestheticMaps\\$new\(\)](#)
- [ThemeAestheticMaps\\$toJson\(\)](#)
- [ThemeAestheticMaps\\$clone\(\)](#)

Method new(): Create a new ThemeAestheticMaps object

Usage:

```
ThemeAestheticMaps$new(
  color = NULL,
  fill = NULL,
  shape = NULL,
  size = NULL,
  linetype = NULL,
  alpha = NULL
)
```

Arguments:

color color map as list, character or numeric vector
 fill fill map as list, character or numeric vector
 shape shape map as list, character or numeric vector
 size size map as list, character or numeric vector
 linetype linetype map as list, character or numeric vector
 alpha alpha map as list, character or numeric vector

Returns: A new ThemeAestheticMaps object

Method toJson(): Translate object into a json list

Usage:

```
ThemeAestheticMaps$toJson()
```

Returns: A list that can be saved into a json file

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

`ThemeAestheticSelections`*ThemeAestheticSelections*

Description

R6 class defining how plot configurations will use aesthetic maps

Super class

`tlf::ThemeAestheticMaps -> ThemeAestheticSelections`

Methods**Public methods:**

- [ThemeAestheticSelections\\$new\(\)](#)
- [ThemeAestheticSelections\\$toJson\(\)](#)
- [ThemeAestheticSelections\\$clone\(\)](#)

Method `new()`: Create a new `ThemeAestheticSelections` object

Usage:

```
ThemeAestheticSelections$new(  
  color = NULL,  
  fill = NULL,  
  shape = NULL,  
  size = NULL,  
  linetype = NULL,  
  alpha = NULL  
)
```

Arguments:

`color` selection key or values for choice of color
`fill` selection key or values for choice of fill
`shape` selection key or values for choice of shape
`size` selection key or values for choice of size
`linetype` selection key or values for choice of linetype
`alpha` selection key or values for choice of alpha

Returns: A new `ThemeAestheticSelections` object

Method `toJson()`: Translate object into a json list

Usage:

```
ThemeAestheticSelections$toJson()
```

Returns: A list that can be saved into a json file

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

 ThemeBackground

ThemeBackground

Description

R6 class defining theme background properties

Public fields

watermark character defining content of watermark
 legendPosition character defining where legend should usually be placed
 legendTitle character defining the content of legend title
 plot BackgroundElement object for plot area properties (outside of panel)
 panel BackgroundElement object for plot area properties (inside of panel)
 xAxis BackgroundElement object for x axis properties
 yAxis BackgroundElement object for y axis properties
 y2Axis BackgroundElement object for right y axis properties
 xGrid BackgroundElement object for x grid properties
 yGrid BackgroundElement object for y grid properties
 y2Grid BackgroundElement object for right y grid properties
 legend BackgroundElement object for legend area properties

Methods**Public methods:**

- [ThemeBackground\\$new\(\)](#)
- [ThemeBackground\\$toJson\(\)](#)
- [ThemeBackground\\$clone\(\)](#)

Method `new()`: Create a new ThemeBackground object

Usage:

```
ThemeBackground$new(
  watermark = NULL,
  legendPosition = NULL,
  legendTitle = NULL,
  plot = NULL,
  panel = NULL,
```

```

    xAxis = NULL,
    yAxis = NULL,
    y2Axis = NULL,
    xGrid = NULL,
    yGrid = NULL,
    y2Grid = NULL,
    legend = NULL,
    baseFill = "white",
    baseColor = "black",
    baseSize = 0.5,
    baseLinetype = "solid"
)

```

Arguments:

watermark character defining content of watermark

legendPosition character defining where legend should usually be placed

legendTitle character defining the content of legend title

plot BackgroundElement object or list for plot area properties (outside of panel)

panel BackgroundElement object or list for plot area properties (inside of panel)

xAxis BackgroundElement object or list for x axis properties

yAxis BackgroundElement object or list for y axis properties

y2Axis BackgroundElement object or list for right y axis properties

xGrid BackgroundElement object or list for x grid properties

yGrid BackgroundElement object or list for y grid properties

y2Grid BackgroundElement object or list for right y grid properties

legend BackgroundElement object or list for legend area properties

baseFill name of base color fill of undefined background elements. Default is "white".

baseColor name of base color of undefined background elements. Default is "black".

baseSize name of base size of undefined background elements. Default is 0.5.

baseLinetype name of base size of undefined background elements. Default is "solid".

Returns: A new ThemeBackground object

Method toJson(): Translate object into a json list

Usage:

```
ThemeBackground$toJson()
```

Returns: A list that can be saved into a json file

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

ThemeFont

*ThemeFont***Description**

R6 class defining theme font properties

Public fields

title Font object for font properties title
 subtitle Font object for font properties of subtitle
 xlabel Font object for font properties of xlabel
 ylabel Font object for font properties of ylabel
 y2label Font object for font properties of y2label
 caption Font object for font properties of caption
 watermark Font object for font properties of watermark
 legendTitle Font object for font properties of legend title
 legend Font object for font properties of legend
 xAxis Font object for font properties of xAxis
 yAxis Font object for font properties of yAxis
 y2Axis Font object for font properties of y2Axis

Methods**Public methods:**

- [ThemeFont\\$new\(\)](#)
- [ThemeFont\\$toJson\(\)](#)
- [ThemeFont\\$clone\(\)](#)

Method `new()`: Create a new ThemeFont object

Usage:

```
ThemeFont$new(
  title = NULL,
  subtitle = NULL,
  xlabel = NULL,
  ylabel = NULL,
  y2label = NULL,
  caption = NULL,
  watermark = NULL,
  legendTitle = NULL,
  legend = NULL,
  xAxis = NULL,
  yAxis = NULL,
```

```

    y2Axis = NULL,
    baseColor = "black",
    baseSize = 12,
    baseFace = "plain",
    baseFamily = "",
    baseAngle = 0,
    baseAlign = "center",
    baseMaxWidth = NULL
  )

```

Arguments:

title Font object or list for font properties title
 subtitle Font object or list for font properties of subtitle
 xlabel Font object or list for font properties of xlabel
 ylabel Font object or list for font properties of ylabel
 y2label Font object or list for font properties of y2label
 caption Font object or list for font properties of caption
 watermark Font object or list for font properties of watermark
 legendTitle Font object or list for font properties of legend title
 legend Font object or list for font properties of legend
 xAxis Font object or list for font properties of xAxis
 yAxis Font object or list for font properties of yAxis
 y2Axis Font object or list for font properties of y2Axis
 baseColor name of base color of undefined fonts. Default is "black".
 baseSize base size of undefined fonts. Default is 12.
 baseFace name of base face of undefined fonts. Default is "plain".
 baseFamily name of base family of undefined fonts. Default is "".
 baseAngle base angle of undefined fonts. Default is 0 degree.
 baseAlign base alignment of undefined fonts. Default is "center".
 baseMaxWidth base maximum ggplot2 "pt" unit in which text is drawn.

Returns: A new ThemeFont object

Method toJson(): Translate object into a json list

Usage:

```
ThemeFont$toJson()
```

Returns: A list that can be saved into a json file

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

ThemePlotConfigurations

ThemePlotConfigurations

Description

R6 class defining theme of plot configuration objects

Public fields

`addScatter` theme properties for PlotConfiguration objects as used in function `addScatter()`

`addLine` theme properties for PlotConfiguration objects as used in function `addLine()`

`addRibbon` theme properties for PlotConfiguration objects as used in function `addRibbon()`

`addErrorbar` theme properties for PlotConfiguration objects as used in function `addErrorbar()`

Methods

Public methods:

- `ThemePlotConfigurations$new()`
- `ThemePlotConfigurations$toJson()`
- `ThemePlotConfigurations$clone()`

Method `new()`: Create a new ThemePlotConfigurations object

Usage:

```
ThemePlotConfigurations$new(  
  addScatter = NULL,  
  addLine = NULL,  
  addRibbon = NULL,  
  addErrorbar = NULL,  
  ...  
)
```

Arguments:

`addScatter` theme properties for PlotConfiguration objects as used in function `addScatter()`

`addLine` theme properties for PlotConfiguration objects as used in function `addLine()`

`addRibbon` theme properties for PlotConfiguration objects as used in function `addRibbon()`

`addErrorbar` theme properties for PlotConfiguration objects as used in function `addErrorbar()`

`...` theme properties for PlotConfiguration objects as used in molecule plots

Returns: A new ThemePlotConfigurations object

Method `toJson()`: Translate object into a json list

Usage:

```
ThemePlotConfigurations$toJson()
```

Returns: A list that can be saved into a json file

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

TickLabelTransforms	<i>TickLabelTransforms</i>
---------------------	----------------------------

Description

List of all available tick label transformation names

Usage

```
TickLabelTransforms
```

Format

An object of class list of length 9.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [VerticalJustification](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

TimeProfileDataMapping	<i>TimeProfileDataMapping</i>
------------------------	-------------------------------

Description

R6 class defining the configuration of a ggplot object for time profile plot

Super classes

```
tlf::XYDataMapping -> tlf::XYGDataMapping -> tlf::RangeDataMapping -> TimeProfileDataMapping
```

Public fields

y2Axis Name of y2Axis variable to map

Methods

Public methods:

- [TimeProfileDataMapping\\$new\(\)](#)
- [TimeProfileDataMapping\\$checkMapData\(\)](#)
- [TimeProfileDataMapping\\$requireDualAxis\(\)](#)
- [TimeProfileDataMapping\\$getLeftAxis\(\)](#)
- [TimeProfileDataMapping\\$getRightAxis\(\)](#)
- [TimeProfileDataMapping\\$clone\(\)](#)

Method `new()`: Create a new `TimeProfileDataMapping` object

Usage:

```
TimeProfileDataMapping$new(
  x = NULL,
  y = NULL,
  ymin = NULL,
  ymax = NULL,
  group = NULL,
  y2Axis = NULL,
  color = NULL,
  fill = NULL,
  linetype = NULL,
  data = NULL
)
```

Arguments:

`x` Name of x variable to map
`y` Name of y variable to map
`ymin` Name of ymin variable to map
`ymax` Name of ymax variable to map
`group` R6 class Grouping object or its input
`y2Axis` Name of y2Axis variable to map
`color` R6 class Grouping object or its input
`fill` R6 class Grouping object or its input
`linetype` R6 class Grouping object or its input
`data` data.frame to map used by `.smartMapping`

Returns: A new `RangeDataMapping` object

Method `checkMapData()`: Check that data variables include map variables

Usage:

```
TimeProfileDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

`data` data.frame to check
`metaData` list containing information on data

Returns: A data.frame with map and legendLabels variables. Dummy variable legendLabels is necessary to allow further modification of plots.

Method requireDualAxis(): Assess if data require a dual axis plot

Usage:

```
TimeProfileDataMapping$requireDualAxis(data)
```

Arguments:

data data.frame to check

Returns: A logical

Method getLeftAxis(): Render NA values for all right axis data

Usage:

```
TimeProfileDataMapping$getLeftAxis(data)
```

Arguments:

data A data.frame

Returns: A data.frame to be plotted in left axis

Method getRightAxis(): Render NA values for all left axis data

Usage:

```
TimeProfileDataMapping$getRightAxis(data)
```

Arguments:

data A data.frame

Returns: A data.frame to be plotted in right axis

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

```
TimeProfilePlotConfiguration
      TimeProfilePlotConfiguration
```

Description

R6 class defining the configuration of a ggplot object for time profile plots

Super class

```
tlf::PlotConfiguration -> TimeProfilePlotConfiguration
```

Public fields

`lloqDirection` Whether to draw LLOQ lines for x (vertical), y (horizontal) or x and y (both).

Active bindings

`y2Axis` YAxisConfiguration object defining properties of y2-axis

Methods

Public methods:

- [TimeProfilePlotConfiguration\\$new\(\)](#)
- [TimeProfilePlotConfiguration\\$clone\(\)](#)

Method `new()`: Create a new TimeProfilePlotConfiguration object

Usage:

```
TimeProfilePlotConfiguration$new(
  ...,
  y2label = NULL,
  y2Axis = NULL,
  y2Scale = NULL,
  y2ValuesLimits = NULL,
  y2AxisLimits = NULL,
  y2Limits = lifecycle::deprecated(),
  lloqDirection = "horizontal",
  data = NULL,
  metaData = NULL,
  dataMapping = NULL
)
```

Arguments:

... parameters inherited from PlotConfiguration
 y2label character or Label object defining plot y2label
 y2Axis YAxisConfiguration object defining y-axis properties

y2Scale name of y2-axis scale. Use enum `Scaling` to access predefined scales.
 y2ValuesLimits numeric vector of length 2 defining y values limits
 y2AxisLimits numeric vector of length 2 defining y axis limits
 y2Limits **[Deprecated]**. Replaced by `y2AxisLimits` argument.
 lloqDirection Whether to draw LLOQ lines for x (vertical), y (horizontal) or x and y (both).
 data data.frame used by `.smartMapping`
 metaData list of information on data
 dataMapping R6 class or subclass `TimeProfileDataMapping`
Returns: A new `TimeProfilePlotConfiguration` object

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `PlotConfiguration` classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

tlfSettingsNames	<i>tlfSettingsNames</i>
------------------	-------------------------

Description

Names of the default/global settings stored in `tlfEnv`. Can be used with `getTLFSettings()`

Usage

```
tlfSettingsNames
```

Format

An object of class `list` of length 14.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfStatFunctions](#)

tlfStatFunctions *tlfStatFunctions*

Description

Bank of predefined functions ready to use by Aggregation methods. Bank defined as Enum. To access the function from its name, use match.fun: e.g. testFun <- match.fun("mean-1.96sd")

Usage

```
tlfStatFunctions
```

Format

An object of class list of length 31.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [VerticalJustification](#), [tlfSettingsNames](#)

TornadoDataMapping *TornadoDataMapping*

Description

R6 class for mapping values, labels to data

Super classes

```
tlf::XYDataMapping -> tlf::XYGDataMapping -> TornadoDataMapping
```

Public fields

lines numeric vector of limits to plot

sorted logical indicating if values should be sorted

Methods

Public methods:

- [TornadoDataMapping\\$new\(\)](#)
- [TornadoDataMapping\\$checkMapData\(\)](#)
- [TornadoDataMapping\\$clone\(\)](#)

Method `new()`: Create a new `TornadoDataMapping` object

Usage:

```
TornadoDataMapping$new(
  lines = DefaultDataMappingValues$tornado,
  sorted = NULL,
  x = NULL,
  y = NULL,
  ...
)
```

Arguments:

`lines` numeric vector of limits to plot
`sorted` logical indicating if values should be sorted
`x` Variable including the values of tornado plot
`y` Variable including the labels of tornado plot
`...` parameters inherited from `XYGDataMapping`

Returns: A new `TornadoDataMapping` object

Method `checkMapData()`: Check that data variables include map variables

Usage:

```
TornadoDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

`data` data.frame to check
`metaData` list containing information on data

Returns: A data.frame with `map` and `defaultAes` variables. Dummy variable `defaultAes` is necessary to allow further modification of plots.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `DataMapping` classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [XYDataMapping](#), [XYGDataMapping](#)

TornadoPlotConfiguration

TornadoPlotConfiguration

Description

R6 class defining the configuration of a ggplot object for tornado plots

Super class

tlf::PlotConfiguration -> TornadoPlotConfiguration

Public fields

bar logical setting if tornado is uses a bar plot instead of regular points

colorPalette color palette property from ggplot2

dodge space between the bars/points

defaultYScale Default yAxis scale value when creating a TornadoPlotConfiguration object

Methods

Public methods:

- [TornadoPlotConfiguration\\$new\(\)](#)
- [TornadoPlotConfiguration\\$clone\(\)](#)

Method new(): Create a new TornadoPlotConfiguration object

Usage:

```
TornadoPlotConfiguration$new(bar = TRUE, colorPalette = NULL, dodge = 0.5, ...)
```

Arguments:

bar logical setting if tornado is uses a bar plot instead of regular points

colorPalette color palette property from ggplot2

dodge space between the bars/points

... parameters inherited from PlotConfiguration

Returns: A new TornadoPlotConfiguration object

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [XAxisConfiguration](#), [YAxisConfiguration](#)

updateExportDimensionsForLegend
<i>updateExportDimensionsForLegend</i>

Description

Update plot dimensions based on size and position of legend

Usage

updateExportDimensionsForLegend(plotObject)

Arguments

plotObject A ggplot object

Value

A ggplot object

updateTimeProfileLegend
<i>updateTimeProfileLegend</i>

Description

Update time profile legend caption

Usage

updateTimeProfileLegend(plotObject, caption)

Arguments

plotObject A ggplot object
caption A data.frame as obtained from getLegendCaption to use for updating a plot legend.

Value

A ggplot object

useDarkTheme	<i>useDarkTheme</i>
--------------	---------------------

Description

Set default Matlab theme to be used as the current default of the tlf environment

Usage

useDarkTheme()

Examples

```
useDarkTheme()
addScatter(x = seq(1, 10), y = rnorm(10))
```

useExcelTheme	<i>useExcelTheme</i>
---------------	----------------------

Description

Set default Excel theme to be used as the current default of the tlf environment

Usage

useExcelTheme()

Examples

```
useExcelTheme()
addScatter(x = seq(1, 10), y = rnorm(10))
```

useHighChartTheme	<i>useHighChartTheme</i>
-------------------	--------------------------

Description

Set default HighChart theme to be used as the current default of the tlf environment

Usage

```
useHighChartTheme()
```

Examples

```
useHighChartTheme()  
addScatter(x = seq(1, 10), y = rnorm(10))
```

useMatlabTheme	<i>useMatlabTheme</i>
----------------	-----------------------

Description

Set default Matlab theme to be used as the current default of the tlf environment

Usage

```
useMatlabTheme()
```

Examples

```
useMatlabTheme()  
addScatter(x = seq(1, 10), y = rnorm(10))
```

useMinimalTheme	<i>useMinimalTheme</i>
-----------------	------------------------

Description

Set default minimal theme to be used as the current default of the tlf environment

Usage

```
useMinimalTheme()
```

Examples

```
useMinimalTheme()  
addScatter(x = seq(1, 10), y = rnorm(10))
```

useTemplateTheme	<i>useTemplateTheme</i>
------------------	-------------------------

Description

Set default theme to be used as the current default of the tlf environment

Usage

```
useTemplateTheme()
```

Examples

```
useTemplateTheme()
addScatter(x = seq(1, 10), y = rnorm(10))
```

useTheme	<i>useTheme</i>
----------	-----------------

Description

set the theme to be used as the current default of the tlf environment

Usage

```
useTheme(theme)
```

Arguments

theme	Theme to be used as default of the tlf environment
-------	--

VerticalJustification	<i>VerticalJustification</i>
-----------------------	------------------------------

Description

List of all available vertical justifications for plot annotation text.

Usage

```
VerticalJustification
```

Format

An object of class list of length 3.

See Also

Other enum helpers: [AestheticFields](#), [AestheticProperties](#), [AestheticSelectionKeys](#), [Alignments](#), [AtomPlots](#), [ColorMaps](#), [ColorPalettes](#), [DataMappings](#), [DefaultDataMappingValues](#), [Directions](#), [ExportFormats](#), [ExportUnits](#), [FontFaces](#), [HorizontalJustification](#), [LegendPositions](#), [LegendTypes](#), [Linetypes](#), [MoleculePlots](#), [PlotAnnotationTextSize](#), [PlotConfigurations](#), [Scaling](#), [Shapes](#), [TagPositions](#), [TickLabelTransforms](#), [tlfSettingsNames](#), [tlfStatFunctions](#)

XAxisConfiguration	<i>XAxisConfiguration</i>
--------------------	---------------------------

Description

R6 class defining the configuration of X-axis

Super class

`tlf::AxisConfiguration -> XAxisConfiguration`

Methods**Public methods:**

- [XAxisConfiguration\\$updatePlot\(\)](#)
- [XAxisConfiguration\\$clone\(\)](#)

Method `updatePlot()`: Update axis configuration on a ggplot object

Usage:

```
XAxisConfiguration$updatePlot(  
  plotObject,  
  yAxisLimits = NULL,  
  ylim = lifecycle::deprecated()  
)
```

Arguments:

`plotObject` ggplot object

`yAxisLimits` values of axisLimits for y axis to prevent `coord_cartesian` to overwrite its properties

`ylim` **[Deprecated]**. Replaced by `yAxisLimits` argument.

Returns: A ggplot object with updated axis properties

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other PlotConfiguration classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [YAxisConfiguration](#)

XYDataMapping

XYDataMapping

Description

R6 class for mapping x and y variable to data

Public fields

x Name of x variable to map
y Name of y variable to map
data data.frame used for mapping

Methods**Public methods:**

- [XYDataMapping\\$new\(\)](#)
- [XYDataMapping\\$checkMapData\(\)](#)
- [XYDataMapping\\$clone\(\)](#)

Method `new()`: Create a new XYDataMapping object

Usage:

```
XYDataMapping$new(x, y = NULL)
```

Arguments:

x Name of x variable to map
y Name of y variable to map

Returns: A new XYDataMapping object

Method `checkMapData()`: Check that data variables include map variables

Usage:

```
XYDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check
metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYGDataMapping](#)

XYGDataMapping

XYGDataMapping

Description

R6 class for mapping x, y and GroupMapping variables to data

Super class

```
tlf::XYDataMapping -> XYGDataMapping
```

Public fields

groupMapping R6 class GroupMapping object

Methods

Public methods:

- [XYGDataMapping\\$new\(\)](#)
- [XYGDataMapping\\$checkMapData\(\)](#)
- [XYGDataMapping\\$clone\(\)](#)

Method new(): Create a new XYGDataMapping object

Usage:

```

XYGDataMapping$new(
  x = NULL,
  y = NULL,
  groupMapping = NULL,
  color = NULL,
  fill = NULL,
  linetype = NULL,
  shape = NULL,
  size = NULL,
  group = NULL,
  data = NULL
)

```

Arguments:

x Name of x variable to map
y Name of y variable to map
groupMapping R6 class GroupMapping object
color R6 class Grouping object or its input
fill R6 class Grouping object or its input
linetype R6 class Grouping object or its input
shape R6 class Grouping object or its input
size R6 class Grouping object or its input
group R6 class Grouping object or its input
data data.frame to map used by .smartMapping

Returns: A new XYGDataMapping object

Method checkMapData(): Check that data variables include map variables

Usage:

```
XYGDataMapping$checkMapData(data, metaData = NULL)
```

Arguments:

data data.frame to check
metaData list containing information on data

Returns: A data.frame with map and defaultAes variables. Dummy variable defaultAes is necessary to allow further modification of plots.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other DataMapping classes: [BoxWhiskerDataMapping](#), [CumulativeTimeProfileDataMapping](#), [DDIRatioDataMapping](#), [GroupMapping](#), [Grouping](#), [HistogramDataMapping](#), [ObsVsPredDataMapping](#), [ObservedDataMapping](#), [PKRatioDataMapping](#), [PieChartDataMapping](#), [QQDataMapping](#), [RangeDataMapping](#), [ResVsPredDataMapping](#), [ResVsTimeDataMapping](#), [TimeProfileDataMapping](#), [TornadoDataMapping](#), [XYDataMapping](#)

YAxisConfiguration	<i>YAxisConfiguration</i>
--------------------	---------------------------

Description

R6 class defining the configuration of Y-axis

Super class

`tlf::AxisConfiguration -> YAxisConfiguration`

Public fields

`position` character position of the Y-axis

Methods**Public methods:**

- [YAxisConfiguration\\$updatePlot\(\)](#)
- [YAxisConfiguration\\$clone\(\)](#)

Method `updatePlot()`: Update axis configuration on a ggplot object

Usage:

```
YAxisConfiguration$updatePlot(
  plotObject,
  xAxisLimits = NULL,
  xlim = lifecycle::deprecated()
)
```

Arguments:

`plotObject` ggplot object

`xAxisLimits` limits of x axis to prevent `coord_cartesian` to overwrite its properties

`xlim` **[Deprecated]**. Replaced by `xAxisLimits` argument.

Returns: A ggplot object with updated axis properties

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other `PlotConfiguration` classes: [AxisConfiguration](#), [BackgroundConfiguration](#), [BackgroundElement](#), [BoxWhiskerPlotConfiguration](#), [CumulativeTimeProfilePlotConfiguration](#), [DDIRatioPlotConfiguration](#), [ExportConfiguration](#), [HistogramPlotConfiguration](#), [LabelConfiguration](#), [LegendConfiguration](#), [LineElement](#), [ObsVsPredPlotConfiguration](#), [PKRatioPlotConfiguration](#), [PieChartPlotConfiguration](#), [PlotConfiguration](#), [PlotGridConfiguration](#), [QQPlotConfiguration](#), [ResVsPredPlotConfiguration](#), [ResVsTimePlotConfiguration](#), [TimeProfilePlotConfiguration](#), [TornadoPlotConfiguration](#), [XAxisConfiguration](#)

Index

* **DataMapping classes**

- BoxWhiskerDataMapping, [28](#)
- CumulativeTimeProfileDataMapping, [33](#)
- DDIRatioDataMapping, [35](#)
- Grouping, [59](#)
- GroupMapping, [60](#)
- HistogramDataMapping, [61](#)
- ObservedDataMapping, [81](#)
- ObsVsPredDataMapping, [83](#)
- PieChartDataMapping, [101](#)
- PKRatioDataMapping, [103](#)
- QQDataMapping, [136](#)
- RangeDataMapping, [138](#)
- ResVsPredDataMapping, [140](#)
- ResVsTimeDataMapping, [141](#)
- TimeProfileDataMapping, [181](#)
- TornadoDataMapping, [186](#)
- XYDataMapping, [194](#)
- XYGDataMapping, [195](#)

* **PlotConfiguration classes**

- AxisConfiguration, [23](#)
- BackgroundConfiguration, [25](#)
- BackgroundElement, [27](#)
- BoxWhiskerPlotConfiguration, [30](#)
- CumulativeTimeProfilePlotConfiguration, [34](#)
- DDIRatioPlotConfiguration, [36](#)
- ExportConfiguration, [38](#)
- HistogramPlotConfiguration, [62](#)
- LabelConfiguration, [68](#)
- LegendConfiguration, [69](#)
- LineElement, [72](#)
- ObsVsPredPlotConfiguration, [85](#)
- PieChartPlotConfiguration, [101](#)
- PKRatioPlotConfiguration, [104](#)
- PlotConfiguration, [107](#)
- PlotGridConfiguration, [115](#)
- QQPlotConfiguration, [137](#)

- ResVsPredPlotConfiguration, [140](#)
- ResVsTimePlotConfiguration, [142](#)
- TimeProfilePlotConfiguration, [184](#)
- TornadoPlotConfiguration, [188](#)
- XAxisConfiguration, [193](#)
- YAxisConfiguration, [197](#)

* **atom plots**

- addErrorbar, [7](#)
- addLine, [9](#)
- addRibbon, [11](#)
- addScatter, [13](#)
- initializePlot, [64](#)

* **datasets**

- AestheticFields, [17](#)
- AestheticProperties, [17](#)
- AestheticSelectionKeys, [18](#)
- Alignments, [21](#)
- AtomPlots, [22](#)
- ColorMaps, [31](#)
- ColorPalettes, [32](#)
- DataMappings, [35](#)
- DefaultDataMappingValues, [37](#)
- Directions, [38](#)
- ExportFormats, [40](#)
- ExportUnits, [42](#)
- FontFaces, [45](#)
- HorizontalJustification, [63](#)
- LegendPositions, [71](#)
- LegendTypes, [71](#)
- Linetypes, [73](#)
- MoleculePlots, [80](#)
- PlotAnnotationTextSize, [105](#)
- PlotConfigurations, [110](#)
- Scaling, [144](#)
- Shapes, [171](#)
- TagPositions, [172](#)
- TickLabelTransforms, [181](#)
- tlfSettingsNames, [185](#)
- tlfStatFunctions, [186](#)

- VerticalJustification, 192
- * **enum helpers**
 - AestheticFields, 17
 - AestheticProperties, 17
 - AestheticSelectionKeys, 18
 - Alignments, 21
 - AtomPlots, 22
 - ColorMaps, 31
 - ColorPalettes, 32
 - DataMappings, 35
 - DefaultDataMappingValues, 37
 - Directions, 38
 - ExportFormats, 40
 - ExportUnits, 42
 - FontFaces, 45
 - HorizontalJustification, 63
 - LegendPositions, 71
 - LegendTypes, 71
 - Linetypes, 73
 - MoleculePlots, 80
 - PlotAnnotationTextSize, 105
 - PlotConfigurations, 110
 - Scaling, 144
 - Shapes, 171
 - TagPositions, 172
 - TickLabelTransforms, 181
 - tlfSettingsNames, 185
 - tlfStatFunctions, 186
 - VerticalJustification, 192
- * **molecule plots**
 - plotBoxWhisker, 105
 - plotCumulativeTimeProfile, 111
 - plotDDIRatio, 112
 - plotGrid, 114
 - plotHistogram, 120
 - plotObservedTimeProfile, 122
 - plotObsVsPred, 123
 - plotPieChart, 125
 - plotPKRatio, 127
 - plotQQ, 128
 - plotResVsPred, 129
 - plotResVsTime, 131
 - plotSimulatedTimeProfile, 132
 - plotTimeProfile, 133
 - plotTornado, 134
- * **shiny apps**
 - runPlotMaker, 143
 - runThemeMaker, 143
- * **stat functions**
 - mean+1.96sd, 76
 - mean+sd, 77
 - mean-1.96sd, 75
 - mean-sd, 75
 - median+1.5IQR, 79
 - median+IQR, 80
 - median-1.5IQR, 78
 - median-IQR, 78
 - Percentile0%, 86
 - Percentile100%, 89
 - Percentile10%, 88
 - Percentile15%, 89
 - Percentile1%, 87
 - Percentile2.5%, 90
 - Percentile20%, 91
 - Percentile25%, 92
 - Percentile25%-1.5IQR, 92
 - Percentile50%, 94
 - Percentile5%, 93
 - Percentile75%, 95
 - Percentile75%+1.5IQR, 95
 - Percentile80%, 96
 - Percentile85%, 97
 - Percentile90%, 98
 - Percentile95%, 98
 - Percentile97.5%, 99
 - Percentile99%, 100
 - .sanitizeLabel, 7
 - addErrorbar, 7, 10, 13, 15, 64
 - addLine, 8, 9, 13, 15, 64
 - addRibbon, 8, 10, 11, 15, 64
 - addScatter, 8, 10, 13, 13, 64
 - addWatermark, 15
 - AestheticFields, 17, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
 - AestheticProperties, 17, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
 - AestheticSelectionKeys, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
 - AggregationInput, 18
 - AggregationSummary, 19

- Alignments, [17](#), [18](#), [21](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#),
[45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- asLabel, [22](#)
- AtomPlots, [17](#), [18](#), [22](#), [22](#), [32](#), [35](#), [38](#), [40](#), [42](#),
[45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- AxisConfiguration, [23](#), [27](#), [28](#), [31](#), [34](#), [37](#),
[40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#),
[119](#), [137](#), [141](#), [142](#), [185](#), [189](#), [194](#),
[198](#)
- BackgroundConfiguration, [25](#), [25](#), [28](#), [31](#),
[34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#),
[105](#), [110](#), [119](#), [137](#), [141](#), [142](#), [185](#),
[189](#), [194](#), [198](#)
- BackgroundElement, [25](#), [27](#), [27](#), [31](#), [34](#), [37](#),
[40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#),
[119](#), [137](#), [141](#), [142](#), [185](#), [189](#), [194](#),
[198](#)
- BoxWhiskerDataMapping, [28](#), [33](#), [36](#), [60–62](#),
[83](#), [85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [142](#),
[183](#), [187](#), [195](#), [197](#)
- BoxWhiskerPlotConfiguration, [25](#), [27](#), [28](#),
[30](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#),
[102](#), [105](#), [110](#), [119](#), [137](#), [141](#), [142](#),
[185](#), [189](#), [194](#), [198](#)
- ColorMaps, [17](#), [18](#), [22](#), [23](#), [31](#), [32](#), [35](#), [38](#), [40](#),
[42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- ColorPalettes, [17](#), [18](#), [22](#), [23](#), [32](#), [32](#), [35](#), [38](#),
[40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#),
[145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- createWatermarkGrob, [32](#)
- CumulativeTimeProfileDataMapping, [30](#),
[33](#), [36](#), [60–62](#), [83](#), [85](#), [101](#), [104](#), [137](#),
[139](#), [140](#), [142](#), [183](#), [187](#), [195](#), [197](#)
- CumulativeTimeProfilePlotConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#),
[72](#), [86](#), [102](#), [105](#), [110](#), [119](#), [137](#), [141](#),
[142](#), [185](#), [189](#), [194](#), [198](#)
- DataMappings, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#),
[42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- DDIRatioDataMapping, [30](#), [33](#), [35](#), [60–62](#), [83](#),
[85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [142](#),
[183](#), [187](#), [195](#), [197](#)
- DDIRatioPlotConfiguration, [25](#), [27](#), [28](#), [31](#),
[34](#), [36](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#),
[105](#), [110](#), [119](#), [137](#), [141](#), [142](#), [185](#),
[189](#), [194](#), [198](#)
- DefaultDataMappingValues, [17](#), [18](#), [22](#), [23](#),
[32](#), [35](#), [37](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#),
[81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#),
[185](#), [186](#), [193](#)
- Directions, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [38](#), [40](#),
[42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- element_text(), [117](#)
- ExportConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#),
[38](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#),
[119](#), [137](#), [141](#), [142](#), [185](#), [189](#), [194](#),
[198](#)
- ExportFormats, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#),
[42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- exportPlot, [41](#)
- exportPlotConfigurationCode, [42](#)
- ExportUnits, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#),
[42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- facet_wrap(), [117](#), [118](#)
- Font, [43](#)
- FontFaces, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#),
[45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#),
[171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- geomTLFPoint, [46](#)
- getBoxWhiskerMeasure, [46](#)
- getDefaultCaptions, [47](#)
- getDualAxisPlot, [48](#)
- getGreekTickLabels, [49](#)
- getGuestValues, [49](#)
- getGuestValuesFromDataMapping, [50](#)
- getLabelWithUnit, [51](#)
- getLegendCaption, [52](#)
- getLegendPosition, [52](#)
- getLinesFromFoldDistance, [53](#)
- getLnTickLabels, [53](#)
- getLogTickLabels, [54](#)
- getPercentileTickLabels, [55](#)
- getPiTickLabels, [55](#)
- getPKRatioMeasure, [56](#)
- getSameLimits, [57](#)

- `getSqrtTickLabels`, 57
- `getSymmetricLimits`, 58
- `getTLFSettings`, 58
- `Grouping`, 30, 33, 36, 59, 61, 62, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197
- `GroupMapping`, 30, 33, 36, 60, 60, 62, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197
- `HistogramDataMapping`, 30, 33, 36, 60, 61, 61, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197
- `HistogramPlotConfiguration`, 25, 27, 28, 31, 34, 37, 40, 62, 69, 70, 72, 86, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 198
- `HorizontalJustification`, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
- `initializePlot`, 8, 10, 13, 15, 64
- `isBetween`, 65
- `Label`, 65
- `LabelConfiguration`, 25, 27, 28, 31, 34, 37, 40, 63, 68, 70, 72, 86, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 198
- `LegendConfiguration`, 25, 27, 28, 31, 34, 37, 40, 63, 69, 69, 72, 86, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 198
- `LegendPositions`, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
- `LegendTypes`, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
- `LineElement`, 25, 27, 28, 31, 34, 37, 40, 63, 69, 70, 72, 86, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 198
- `Linetypes`, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 81, 105, 110, 145, 171, 172, 181, 185, 186, 193
- `loadThemeFromJson`, 74
- `loadTLFSettings`, 74
- `matrix()`, 117
- `mean+1.96sd`, 75, 76, 76–80, 87–100
- `mean+sd`, 75, 76, 77, 78–80, 87–100
- `mean-1.96sd`, 75
- `mean-sd`, 75
- `median+1.5IQR`, 75–78, 79, 79, 80, 87–100
- `median+IQR`, 75–79, 80, 87–100
- `median-1.5IQR`, 78
- `median-IQR`, 78
- `MoleculePlots`, 17, 18, 22, 23, 32, 35, 38, 40, 42, 45, 63, 71, 73, 80, 105, 110, 145, 171, 172, 181, 185, 186, 193
- `ObservedDataMapping`, 30, 33, 36, 60–62, 81, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197
- `ObsVsPredDataMapping`, 30, 33, 36, 60–62, 83, 83, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197
- `ObsVsPredPlotConfiguration`, 25, 27, 28, 31, 34, 37, 40, 63, 69, 70, 72, 85, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 198
- `ospsuite.utils::Printable`, 116
- `patchwork::area()`, 117
- `Percentile0%`, 75–80, 86, 88–100
- `Percentile100%`, 75–80, 87, 88, 89, 90–100
- `Percentile10%`, 75–80, 87, 88, 88–100
- `Percentile15%`, 75–80, 87, 88, 89, 89–100
- `Percentile1%`, 75–80, 87, 87–100
- `Percentile2.5%`, 75–80, 87–89, 90, 90–100
- `Percentile20%`, 75–80, 87–90, 91, 92–100
- `Percentile25%`, 75–80, 87–91, 92, 92–100
- `Percentile25%-1.5IQR`, 92
- `Percentile50%`, 75–80, 87–93, 94, 95–100
- `Percentile5%`, 75–80, 87–92, 93, 93–100
- `Percentile75%`, 75–80, 87–94, 95, 95–100
- `Percentile75%+1.5IQR`, 75–80, 87–94, 95, 96–100
- `Percentile80%`, 75–80, 87–95, 96, 96–100
- `Percentile85%`, 75–80, 87–96, 97, 98–100
- `Percentile90%`, 75–80, 87–97, 98, 99, 100
- `Percentile95%`, 75–80, 87–97, 98, 98–100
- `Percentile97.5%`, 75–80, 87–98, 99, 99, 100
- `Percentile99%`, 75–80, 87–99, 100
- `PieChartDataMapping`, 30, 33, 36, 60–62, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 197

- PieChartPlotConfiguration, [25](#), [27](#), [28](#), [31](#),
[34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [101](#),
[105](#), [110](#), [119](#), [137](#), [141](#), [142](#), [185](#),
[189](#), [194](#), [198](#)
- PKRatioDataMapping, [30](#), [33](#), [36](#), [60–62](#), [83](#),
[85](#), [101](#), [103](#), [137](#), [139](#), [140](#), [142](#),
[183](#), [187](#), [195](#), [197](#)
- PKRatioPlotConfiguration, [25](#), [27](#), [28](#), [31](#),
[34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#),
[104](#), [110](#), [119](#), [137](#), [141](#), [142](#), [185](#),
[189](#), [194](#), [198](#)
- PlotAnnotationTextSize, [17](#), [18](#), [22](#), [23](#), [32](#),
[35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#),
[105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#),
[186](#), [193](#)
- plotBoxWhisker, [105](#), [111](#), [113](#), [114](#), [121](#),
[123](#), [124](#), [126](#), [128–131](#), [133–135](#)
- PlotConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#),
[40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [107](#),
[119](#), [137](#), [141](#), [142](#), [185](#), [189](#), [194](#),
[198](#)
- PlotConfigurations, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#),
[38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#),
[110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#),
[193](#)
- plotCumulativeTimeProfile, [106](#), [111](#), [113](#),
[114](#), [121](#), [123](#), [124](#), [126](#), [128–131](#),
[133–135](#)
- plotDDIRatio, [106](#), [111](#), [112](#), [114](#), [121](#), [123](#),
[124](#), [126](#), [128–131](#), [133–135](#)
- plotGrid, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#), [124](#),
[126](#), [128–131](#), [133–135](#)
- PlotGridConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#),
[37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#),
[110](#), [115](#), [137](#), [141](#), [142](#), [185](#), [189](#),
[194](#), [198](#)
- plotHistogram, [106](#), [111](#), [113](#), [114](#), [120](#), [123](#),
[124](#), [126](#), [128–131](#), [133–135](#)
- plotObservedTimeProfile, [106](#), [111](#), [113](#),
[114](#), [121](#), [122](#), [124](#), [126](#), [128–131](#),
[133–135](#)
- plotObsVsPred, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[123](#), [126](#), [128–131](#), [133–135](#)
- plotPieChart, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[124](#), [125](#), [128–131](#), [133–135](#)
- plotPKRatio, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[124](#), [126](#), [127](#), [129–131](#), [133–135](#)
- plotQQ, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#), [124](#),
[126](#), [128](#), [128](#), [130](#), [131](#), [133–135](#)
- plotResVsPred, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[124](#), [126](#), [128](#), [129](#), [129](#), [131](#),
[133–135](#)
- plotResVsTime, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[124](#), [126](#), [128–130](#), [131](#), [133–135](#)
- plotSimulatedTimeProfile, [106](#), [111](#), [113](#),
[114](#), [121](#), [123](#), [124](#), [126](#), [128–131](#),
[132](#), [134](#), [135](#)
- plotTimeProfile, [106](#), [111](#), [113](#), [114](#), [121](#),
[123](#), [124](#), [126](#), [128–131](#), [133](#), [133](#),
[135](#)
- plotTornado, [106](#), [111](#), [113](#), [114](#), [121](#), [123](#),
[124](#), [126](#), [128–131](#), [133](#), [134](#), [134](#)
- QQDataMapping, [30](#), [33](#), [36](#), [60–62](#), [83](#), [85](#),
[101](#), [104](#), [136](#), [139](#), [140](#), [142](#), [183](#),
[187](#), [195](#), [197](#)
- QQPlotConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#),
[40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#),
[119](#), [137](#), [141](#), [142](#), [185](#), [189](#), [194](#),
[198](#)
- RangeDataMapping, [30](#), [33](#), [36](#), [60–62](#), [83](#), [85](#),
[101](#), [104](#), [137](#), [138](#), [140](#), [142](#), [183](#),
[187](#), [195](#), [197](#)
- resetTLFSettingsToDefault, [139](#)
- ResVsPredDataMapping, [30](#), [33](#), [36](#), [60–62](#),
[83](#), [85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [142](#),
[183](#), [187](#), [195](#), [197](#)
- ResVsPredPlotConfiguration, [25](#), [27](#), [28](#),
[31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#),
[102](#), [105](#), [110](#), [119](#), [137](#), [140](#), [142](#),
[185](#), [189](#), [194](#), [198](#)
- ResVsTimeDataMapping, [30](#), [33](#), [36](#), [60–62](#),
[83](#), [85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [141](#),
[183](#), [187](#), [195](#), [197](#)
- ResVsTimePlotConfiguration, [25](#), [27](#), [28](#),
[31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#),
[102](#), [105](#), [110](#), [119](#), [137](#), [141](#), [142](#),
[185](#), [189](#), [194](#), [198](#)
- runPlotMaker, [143](#), [143](#)
- runThemeMaker, [143](#), [143](#)
- saveThemeToJson, [144](#)
- saveTLFSettings, [144](#)
- Scaling, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#),
[63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [144](#), [171](#),
[172](#), [181](#), [185](#), [186](#), [193](#)

- setBackground, [145](#)
- setBackgroundPanelArea, [146](#)
- setBackgroundPlotArea, [147](#)
- setCaptionColor, [148](#)
- setCaptionFill, [149](#)
- setCaptionLabels, [149](#)
- setCaptionLinetype, [150](#)
- setCaptionOrder, [150](#)
- setCaptionShape, [151](#)
- setCaptionSize, [151](#)
- setCaptionVisibility, [152](#)
- setDefaultAggregationBins, [152](#)
- setDefaultAggregationFunctions, [153](#)
- setDefaultAggregationLabels, [153](#)
- setDefaultAlphaRatio, [154](#)
- setDefaultErrorbarCapSize, [154](#)
- setDefaultExportParameters, [154](#)
- setDefaultLegendPosition, [155](#)
- setDefaultLegendTitle, [156](#)
- setDefaultLLOQLinetype, [156](#)
- setDefaultLogTicks, [156](#)
- setDefaultMaxCharacterWidth, [157](#)
- setDefaultWatermark, [157](#)
- setGrid, [158](#)
- setLegend, [158](#)
- setLegendCaption, [159](#)
- setLegendFont, [160](#)
- setLegendPosition, [160](#)
- setLegendTitle, [161](#)
- setPlotExport, [161](#)
- setPlotExportDimensions, [162](#)
- setPlotExportFormat, [163](#)
- setPlotExportSize, [163](#)
- setPlotLabels, [164](#)
- setWatermark, [165](#)
- setXAxis, [166](#)
- setXGrid, [167](#)
- setY2Axis, [168](#)
- setYAxis, [169](#)
- setYGrid, [170](#)
- Shapes, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- TagPositions, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- Theme, [172](#)
- theme(legend.position=...), [117](#)
- ThemeAestheticMaps, [173](#)
- ThemeAestheticSelections, [175](#)
- ThemeBackground, [176](#)
- ThemeFont, [178](#)
- ThemePlotConfigurations, [180](#)
- TickLabelTransforms, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- TimeProfileDataMapping, [30](#), [33](#), [36](#), [60–62](#), [83](#), [85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [142](#), [181](#), [187](#), [195](#), [197](#)
- TimeProfilePlotConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#), [119](#), [137](#), [141](#), [142](#), [184](#), [189](#), [194](#), [198](#)
- tlfSettingsNames, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- tlfStatFunctions, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [193](#)
- TornadoDataMapping, [30](#), [33](#), [36](#), [60–62](#), [83](#), [85](#), [101](#), [104](#), [137](#), [139](#), [140](#), [142](#), [183](#), [186](#), [195](#), [197](#)
- TornadoPlotConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#), [119](#), [137](#), [141](#), [142](#), [185](#), [188](#), [194](#), [198](#)
- updateExportDimensionsForLegend, [189](#)
- updateTimeProfileLegend, [189](#)
- useDarkTheme, [190](#)
- useExcelTheme, [190](#)
- useHighChartTheme, [191](#)
- useMatlabTheme, [191](#)
- useMinimalTheme, [191](#)
- useTemplateTheme, [192](#)
- useTheme, [192](#)
- VerticalJustification, [17](#), [18](#), [22](#), [23](#), [32](#), [35](#), [38](#), [40](#), [42](#), [45](#), [63](#), [71](#), [73](#), [81](#), [105](#), [110](#), [145](#), [171](#), [172](#), [181](#), [185](#), [186](#), [192](#)
- wrap_plots(), [117](#)
- XAxisConfiguration, [25](#), [27](#), [28](#), [31](#), [34](#), [37](#), [40](#), [63](#), [69](#), [70](#), [72](#), [86](#), [102](#), [105](#), [110](#)

119, 137, 141, 142, 185, 189, 193, 198

XYDataMapping, *30, 33, 36, 60–62, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 194, 197*

XYGDataMapping, *30, 33, 36, 60–62, 83, 85, 101, 104, 137, 139, 140, 142, 183, 187, 195, 195*

YAxisConfiguration, *25, 27, 28, 31, 34, 37, 40, 63, 69, 70, 72, 86, 102, 105, 110, 119, 137, 141, 142, 185, 189, 194, 197*