

Package: rtemis.dplot3 (via r-universe)

October 13, 2026

Title Interactive 'plotly' Graphics for the 'rtemis' Ecosystem

Version 0.1.0

Date 2026-10-13

Description Interactive 'plotly', 'networkD3', 'threejs' and 'leaflet' graphics: scatter, 3D scatter, bar, box, density, heatmap, pie, calibration, confusion matrix, ROC, survival, time series, spectrogram, volcano, protein annotation, network, map and table plots, with 'rtemis' themes and palettes. These functions began as part of the 'rtemis' package ([doi:10.32614/CRAN.package.rtemis](https://doi.org/10.32614/CRAN.package.rtemis)); 'rtemis.draw' provides their successors.

License BSD_3_clause + file LICENSE

URL <https://www.rtemis.org>,
<https://github.com/rtemis-org/rtemis.dplot3>

BugReports <https://github.com/rtemis-org/rtemis.dplot3/issues>

Depends R (>= 4.4.0)

Imports data.table, grDevices, graphics, plotly, rtemis (>= 1.4.2),
rtemis.core (>= 0.4.6), rtemis.draw (>= 0.5.3), S7, stats,
utils

Suggests colorspace, dendextend (>= 0.18.0), geojsonio, ggplot2,
heatmaply, htmltools, htmlwidgets, igraph, jsonlite, leaflet,
networkD3, pROC, reticulate, seqinr, sf, survival, testthat (>= 3.0.0), threejs, withr

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

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

Repository <https://rtemis-org.r-universe.dev>

Date/Publication 2026-10-13 22:21:42 UTC

RemoteUrl <https://github.com/rtemis-org/rtemis.dplot3>

RemoteRef HEAD

RemoteSha 30dc29185869c683e45fe8c0b9b186332555aed9

Contents

available_draw	3
available_themes	3
choose_theme	4
col2grayscale	5
color_adjust	5
draw_3Dscatter	6
draw_bar	9
draw_box	12
draw_calibration	17
draw_confusion	19
draw_dist	20
draw_fit	24
draw_graphD3	25
draw_graphjs	26
draw_heatmap	28
draw_leaflet	31
draw_pie	33
draw_protein	35
draw_pvals	41
draw_roc	42
draw_scatter	44
draw_spectrogram	49
draw_survfit	52
draw_table	54
draw_ts	55
draw_varimp	58
draw_volcano	59
draw_xt	63
get_palette	67
previewcolor	68
theme_black	69
uniprot_get	83
xt_example	84
xtdescribe	85

Index	86
--------------	-----------

available_draw	<i>Available Draw Functions</i>
----------------	---------------------------------

Description

Print available draw functions for visualization.

Usage

```
available_draw(verbosity = 1L)
```

Arguments

verbosity Integer: Verbosity level.

Value

Named list of draw function descriptions, invisibly.

Author(s)

EDG

Examples

```
available_draw()
```

available_themes	<i>Print available rtemis themes</i>
------------------	---

Description

Print available **rtemis** themes

Usage

```
available_themes()
```

Value

Called for its side effect of printing available themes.

Author(s)

EDG

Examples

```
available_themes()
```

choose_theme

*Select an rtemis theme***Description**

Select an rtemis theme

Usage

```
choose_theme(
  x = c("white", "whitegrid", "whiteigrid", "black", "blackgrid", "blackigrid",
        "darkgray", "darkgraygrid", "darkgrayigrid", "lightgraygrid", "mediumgraygrid"),
  override = NULL
)
```

Arguments

x	Character: Name of theme to select. If not defined, will use <code>getOption("rtemis_theme", "whitegrid")</code> .
override	Optional List: Theme parameters to override defaults.

Details

If x is not defined, `choose_theme()` will use `getOption("rtemis_theme", "whitegrid")` to select the theme. This allows users to set a default theme for all rtemis plots by setting `options(rtemis_theme = "theme_name")` at any point.

Value

Theme object.

Author(s)

EDG

Examples

```
# Get default theme set by options(rtemis_theme = "theme_name").
# If not set, defaults to "whitegrid":
choose_theme()
# Get darkgraygrid theme. Same as `theme_darkgraygrid()`:
choose_theme("darkgraygrid")
# This will use the default theme, and override the foreground color to red:
choose_theme(override = list(fg = "#ff0000"))
```

col2grayscale	<i>Color to Grayscale</i>
---------------	---------------------------

Description

Convert a color to grayscale

Usage

```
col2grayscale(x, what = c("color", "decimal"))
```

Arguments

x	Color to convert to grayscale
what	Character: "color" returns a hexadecimal color, "decimal" returns a decimal between 0 and 1

Details

Uses the NTSC grayscale conversion: $0.299 * R + 0.587 * G + 0.114 * B$

Value

Character: color hex code.

Author(s)

EDG

Examples

```
col2grayscale("red")
col2grayscale("red", "dec")
```

color_adjust	<i>Adjust HSV Color</i>
--------------	-------------------------

Description

Modify alpha, hue, saturation and value (HSV) of a color

Usage

```
color_adjust(color, alpha = NULL, hue = 0, sat = 0, val = 0)
```

Arguments

color	Input color. Any format that <code>grDevices::col2rgb()</code> recognizes
alpha	Numeric: Scale alpha by this amount. Future: replace with absolute setting
hue	Float: How much hue to add to color
sat	Float: How much saturation to add to color
val	Float: How much to increase value of color by

Value

Adjusted color

Author(s)

EDG

Examples

```
previewcolor(c(teal = "#00ffff", teal50 = color_adjust("#00ffff", alpha = 0.5)))
```

draw_3Dscatter

Interactive 3D Scatter Plots

Description

Draw interactive 3D scatter plots using plotly.

Usage

```
draw_3Dscatter(
  x,
  y = NULL,
  z = NULL,
  fit = NULL,
  clustering_config = NULL,
  group = NULL,
  rsq = TRUE,
  mode = "markers",
  order_on_x = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  zlab = NULL,
  alpha = 0.8,
  bg = NULL,
  plot_bg = NULL,
  theme = choose_theme(getOption("rtemis_theme")),
```

```

palette = get_palette(getOption("rtemis_palette")),
axes_square = FALSE,
group_names = NULL,
font_size = 16,
marker_col = NULL,
marker_size = 8,
fit_col = NULL,
fit_alpha = 0.7,
fit_lwd = 2.5,
tick_font_size = 12,
spike_col = NULL,
legend = NULL,
legend_xy = c(0, 1),
legend_xanchor = "left",
legend_yanchor = "auto",
legend_orientation = "v",
legend_col = NULL,
legend_bg = "#FFFFFF00",
legend_border_col = "#FFFFFF00",
legend_borderwidth = 0,
legend_group_gap = 0,
margin = list(t = 30, b = 0, l = 0, r = 0),
fit_params = NULL,
width = NULL,
height = NULL,
padding = 0,
displayModeBar = TRUE,
modeBar_file_format = "svg",
verbosity = 0L,
filename = NULL,
file_width = 500,
file_height = 500,
file_scale = 1
)

```

Arguments

<code>x</code>	Numeric, vector/data.frame/list: x-axis data.
<code>y</code>	Numeric, vector/data.frame/list: y-axis data.
<code>z</code>	Numeric, vector/data.frame/list: z-axis data.
<code>fit</code>	Character: Fit method.
<code>clustering_config</code>	ClusteringConfig object: If provided, cluster on x, y, and z and pass clusters to group.
<code>group</code>	Factor: Grouping variable.
<code>rsq</code>	Logical: If TRUE, print R-squared values in legend if fit is set.
<code>mode</code>	Character, vector: "markers", "lines", "markers+lines".

<code>order_on_x</code>	Logical: If TRUE, order x, y, and z on x.
<code>main</code>	Character: Main title.
<code>xlab</code>	Character: x-axis label.
<code>ylab</code>	Character: y-axis label.
<code>zlab</code>	Character: z-axis label.
<code>alpha</code>	Numeric: Alpha for markers.
<code>bg</code>	Background color.
<code>plot_bg</code>	Plot background color.
<code>theme</code>	Theme object.
<code>palette</code>	Character vector: Colors to use.
<code>axes_square</code>	Logical: If TRUE, draw a square plot.
<code>group_names</code>	Character: Names for groups.
<code>font_size</code>	Numeric: Font size.
<code>marker_col</code>	Color for markers.
<code>marker_size</code>	Numeric: Marker size.
<code>fit_col</code>	Color for fit line.
<code>fit_alpha</code>	Numeric: Alpha for fit surface.
<code>fit_lwd</code>	Numeric: Line width for fit line.
<code>tick_font_size</code>	Numeric: Tick font size.
<code>spike_col</code>	Spike lines color.
<code>legend</code>	Logical: If TRUE, draw legend.
<code>legend_xy</code>	Numeric: Position of legend.
<code>legend_xanchor</code>	Character: X anchor for legend.
<code>legend_yanchor</code>	Character: Y anchor for legend.
<code>legend_orientation</code>	Character: Orientation of legend.
<code>legend_col</code>	Color for legend text.
<code>legend_bg</code>	Color for legend background.
<code>legend_border_col</code>	Color for legend border.
<code>legend_borderwidth</code>	Numeric: Border width for legend.
<code>legend_group_gap</code>	Numeric: Gap between legend groups.
<code>margin</code>	Numeric, named list: Margins for top, bottom, left, right.
<code>fit_params</code>	Hyperparameters for fit.
<code>width</code>	Numeric: Width of plot.
<code>height</code>	Numeric: Height of plot.

padding	Numeric: Graph padding.
displayModeBar	Logical: If TRUE, display mode bar.
modeBar_file_format	Character: File format for mode bar.
verbosity	Integer: Verbosity level.
filename	Character: Filename to save plot.
file_width	Numeric: Width of saved file.
file_height	Numeric: Height of saved file.
file_scale	Numeric: Scale of saved file.

Details

See docs.rtemis.org/r for detailed documentation.

Note that draw_3Dscatter uses the theme's plot_bg as grid_col.

Value

A plotly object.

Author(s)

EDG

Examples

```
draw_3Dscatter(iris, group = iris[["Species"]], theme = theme_darkgraygrid())
```

draw_bar

Interactive Barplots

Description

Draw interactive barplots using plotly

Usage

```
draw_bar(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  alpha = 1,
  horizontal = FALSE,
  theme = choose_theme(getOption("rtemis_theme")),
  palette = get_palette(getOption("rtemis_palette")),
```

```

barmode = c("group", "relative", "stack", "overlay"),
group_names = NULL,
order_by_val = FALSE,
ylim = NULL,
hovernames = NULL,
feature_names = NULL,
font_size = 16,
annotate = FALSE,
annotate_col = theme[["labs_col"]],
legend = NULL,
legend_col = NULL,
legend_xy = c(1, 1),
legend_orientation = "v",
legend_xanchor = "left",
legend_yanchor = "auto",
hline = NULL,
hline_col = NULL,
hline_width = 1,
hline_dash = "solid",
hline_annotate = NULL,
hline_annotation_x = 1,
margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
automargin_x = TRUE,
automargin_y = TRUE,
padding = 0,
displayModeBar = TRUE,
modeBar_file_format = "svg",
filename = NULL,
file_width = 500,
file_height = 500,
file_scale = 1,
verbosity = 0L
)

```

Arguments

<code>x</code>	vector (possibly named), matrix, or data.frame: If matrix or data.frame, rows are groups (can be 1 row), columns are features
<code>main</code>	Character: Main plot title.
<code>xlab</code>	Character: x-axis label.
<code>ylab</code>	Character: y-axis label.
<code>alpha</code>	Float (0, 1]: Transparency for bar colors.
<code>horizontal</code>	Logical: If TRUE, plot bars horizontally
<code>theme</code>	Theme object.
<code>palette</code>	Character vector: Colors to use.

barmode	Character: Type of bar plot to make: "group", "relative", "stack", "overlay". Use "relative" for stacked bars, which handles negative values correctly, unlike "stack", as of writing.
group_names	Character, vector, length = NROW(x): Group names. If NULL, uses rownames(x).
order_by_val	Logical: If TRUE, order bars by increasing value. Only use for single group data.
ylim	Float, vector, length 2: y-axis limits.
hovernames	Character, vector: Optional character vector to show on hover over each bar.
feature_names	Character, vector, length = NCOL(x): Feature names. If NULL, uses colnames(x).
font_size	Float: Font size for all labels.
annotate	Logical: If TRUE, annotate stacked bars
annotate_col	Color for annotations
legend	Logical: If TRUE, draw legend. If NULL, turned on if there is more than one feature present.
legend_col	Color: Legend text color. If NULL, determined by the theme.
legend_xy	Numeric, vector, length 2: x and y for plotly's legend
legend_orientation	"v" or "h" for vertical or horizontal
legend_xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend_yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
hline	Float: If defined, draw a horizontal line at this y value.
hline_col	Color for hline.
hline_width	Float: Width for hline.
hline_dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
hline_annotate	Character: Text of horizontal line annotation if hline is set
hline_annotation_x	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area
margin	Named list: plot margins.
automargin_x	Logical: If TRUE, automatically set x-axis margins
automargin_y	Logical: If TRUE, automatically set y-axis margins
padding	Integer: N pixels to pad plot.
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar_file_format	Character: "svg", "png", "jpeg", "pdf" / any output file type supported by plotly and your system
filename	Character: Path to file to save static plot.
file_width	Integer: File width in pixels for when filename is set.
file_height	Integer: File height in pixels for when filename is set.
file_scale	Numeric: If saving to file, scale plot by this number
verbosity	Integer: Verbosity level.

Details

See docs.rtemis.org/r for detailed documentation.

Value

plotly object.

Author(s)

EDG

Examples

```
draw_bar(VADeaths, legend_xy = c(0, 1))
draw_bar(VADeaths, legend_xy = c(1, 1), legend_xanchor = "left")
# simple individual bars
a <- c(4, 7, 2)
draw_bar(a)
# if input is a data.frame, each row is a group and each column is a feature
b <- data.frame(x = c(3, 5, 7), y = c(2, 1, 8), z = c(4, 5, 2))
rownames(b) <- c("Jen", "Ben", "Ren")
draw_bar(b)
# stacked
draw_bar(b, barmode = "stack")
```

draw_box

Interactive Boxplots & Violin plots

Description

Draw interactive boxplots or violin plots using **plotly**

Usage

```
draw_box(
  x,
  time = NULL,
  time_bin = c("year", "quarter", "month", "day"),
  type = c("box", "violin"),
  group = NULL,
  x_transform = c("none", "scale", "minmax"),
  main = NULL,
  xlab = "",
  ylab = NULL,
  alpha = 0.6,
  bg = NULL,
  plot_bg = NULL,
```

```

theme = choose_theme(getOption("rtemis_theme")),
palette = get_palette(getOption("rtemis_palette")),
boxpoints = "outliers",
quartilemethod = "linear",
xlim = NULL,
ylim = NULL,
violin_box = TRUE,
orientation = "v",
annotate_n = FALSE,
annotate_n_y = 1,
annotate_mean = FALSE,
annotate_meansd = FALSE,
annotate_meansd_y = 1,
annotate_col = theme[["labs_col"]],
xnames = NULL,
group_lines = FALSE,
group_lines_dash = "dot",
group_lines_col = NULL,
group_lines_alpha = 0.5,
labelify = TRUE,
order_by_fn = NULL,
font_size = 16,
ylab_standoff = 18,
legend = NULL,
legend_col = NULL,
legend_xy = NULL,
legend_orientation = "v",
legend_xanchor = "auto",
legend_yanchor = "auto",
xaxis_type = "category",
cataxis_tickangle = "auto",
margin = list(b = 65, l = 65, t = 50, r = 12, pad = 0),
automargin_x = TRUE,
automargin_y = TRUE,
boxgroupgap = NULL,
hovertext = NULL,
show_n = FALSE,
pvals = NULL,
htest = "none",
htest_compare = 0,
htest_y = NULL,
htest_annotate = TRUE,
htest_annotate_x = 0,
htest_annotate_y = -0.065,
htest_star_col = theme[["labs_col"]],
htest_bracket_col = theme[["labs_col"]],
starbracket_pad = c(0.04, 0.05, 0.09),
use_plotly_group = FALSE,

```

```

width = NULL,
height = NULL,
displayModeBar = TRUE,
modeBar_file_format = "svg",
filename = NULL,
file_width = 500,
file_height = 500,
file_scale = 1,
mathjax = NULL
)

```

Arguments

x	Vector or List of vectors: Input
time	Date or date-time vector
time_bin	Character: "year", "quarter", "month", or "day". Period to bin by
type	Character: "box" or "violin"
group	Factor to group by
x_transform	Character: "none", "scale", or "minmax" to use raw values, scaled and centered values or min-max normalized to 0-1, respectively. Transform is applied to each variable before grouping, so that groups are comparable
main	Character: Plot title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
alpha	Float (0, 1]: Transparency for box colors.
bg	Color: Background color.
plot_bg	Color: Background color for plot area.
theme	Theme object.
palette	Character vector: Colors to use.
boxpoints	Character or FALSE: "all", "suspectedoutliers", "outliers".
quartilemethod	Character: "linear", "exclusive", "inclusive". See https://plotly.com/r/box-plots/#choosing-the-algorithm-for-computing-quartiles .
xlim	Numeric vector: x-axis limits
ylim	Numeric vector: y-axis limits
violin_box	Logical: If TRUE and type is "violin" show box within violin plot
orientation	Character: "v" or "h" for vertical, horizontal
annotate_n	Logical: If TRUE, annotate with N in each box
annotate_n_y	Numeric: y position for annotate_n
annotate_mean	Logical: If TRUE, annotate with mean of each box
annotate_meansd	Logical: If TRUE, annotate with mean (SD) of each box

annotate_meansd_y	Numeric: y position for annotate_meansd
annotate_col	Color for annotations
xnames	Character, vector, length = NROW(x): x-axis names. If NULL, tries to set names automatically.
group_lines	Logical: If TRUE, add separating lines between groups of boxplots
group_lines_dash	Character: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot"
group_lines_col	Color for group_lines
group_lines_alpha	Numeric: transparency for group_lines_col
labelify	Logical: If TRUE, labelify x names
order_by_fn	Function: If defined, order boxes by increasing value of this function (e.g. median).
font_size	Float: Font size for all labels.
ylab_standoff	Numeric: Standoff for y-axis label
legend	Logical: If TRUE, draw legend.
legend_col	Color: Legend text color. If NULL, determined by the theme.
legend_xy	Float, vector, length 2: Relative x, y position for legend.
legend_orientation	"v" or "h" for vertical, horizontal
legend_xanchor	Character: Legend's x anchor: "left", "center", "right", "auto"
legend_yanchor	Character: Legend's y anchor: "top", "middle", "bottom", "auto"
xaxis_type	Character: "linear", "log", "date", "category", "multicategory"
cataxis_tickangle	Numeric: Angle for categorical axis tick labels
margin	Named list: plot margins.
automargin_x	Logical: If TRUE, automatically set x-axis margins
automargin_y	Logical: If TRUE, automatically set y-axis margins
boxgroupgap	Numeric: Sets the gap (in plot fraction) between boxes of the same location coordinate
hovertext	Character vector: Text to show on hover for each data point
show_n	Logical: If TRUE, show N in each box
pvals	Numeric vector: Precomputed p-values. Should correspond to each box. Bypasses htest and htest_compare. Requires group to be set
htest	Character: e.g. "t.test", "wilcox.test" to compare each box to the <i>first</i> box. If grouped, compare within each group to the first box. If p-value of test is less than htest.thresh, add asterisk above/ to the side of each box
htest_compare	Integer: 0: Compare all distributions against the first one; 2: Compare every second box to the one before it. Requires group to be set

htest_y	Numeric: y coordinate for htest annotation
htest_annotate	Logical: if TRUE, include htest annotation
htest_annotate_x	Numeric: x-axis paper coordinate for htest annotation
htest_annotate_y	Numeric: y-axis paper coordinate for htest annotation
htest_star_col	Color for htest annotation stars
htest_bracket_col	Color for htest annotation brackets
starbracket_pad	Numeric: Padding for htest annotation brackets
use_plotly_group	If TRUE, use plotly's group arg to group boxes.
width	Numeric: Force plot size to this width. If NULL, fill available space.
height	Numeric: Force plot size to this height. If NULL, fill available space.
displayModeBar	Logical: If TRUE, show plotly's modebar
modeBar_file_format	Character: "svg", "png", "jpeg", "pdf"
filename	Character: Path to file to save static plot.
file_width	Integer: File width in pixels for when filename is set.
file_height	Integer: File height in pixels for when filename is set.
file_scale	Numeric: If saving to file, scale plot by this number
mathjax	Optional Character {"local", "cdn"}: Whether to use local or CDN version of MathJax for rendering mathematical annotations.

Details

See docs.rtemis.org/r for detailed documentation.

For multiple box plots, the recommendation is:

- `x=dat[, columnindex]` for multiple variables of a `data.frame`
- `x=list(a=..., b=..., etc.)` for multiple variables of potentially different length
- `x=split(var, group)` for one variable with multiple groups: group names appear below boxplots
- `x=dat[, columnindex], group = factor` for grouping multiple variables: group names appear in legend

If `orientation == "h"`, `xlab` is applied to y-axis and vice versa. Similarly, `xaxis_type` applies to the y-axis - the "category" type would not normally need changing.

Value

plotly object.

Author(s)

EDG

Examples

```
# A.1 Box plot of 4 variables
draw_box(iris[, 1:4])
# A.2 Grouped Box plot
draw_box(iris[, 1:4], group = iris[["Species"]])
draw_box(iris[, 1:4], group = iris[["Species"]], annotate_n = TRUE)
# B. Boxplot binned by time periods
# Synthetic data with an instantenous shift in distributions
set.seed(2021)
dat1 <- data.frame(alpha = rnorm(200, 0), beta = rnorm(200, 2), gamma = rnorm(200, 3))
dat2 <- data.frame(alpha = rnorm(200, 5), beta = rnorm(200, 8), gamma = rnorm(200, -3))
x <- rbind(dat1, dat2)
startDate <- as.Date("2019-12-04")
endDate <- as.Date("2021-03-31")
time <- seq(startDate, endDate, length.out = 400)
draw_box(x[, 1], time, "year", ylab = "alpha")
draw_box(x, time, "year", legend.xy = c(0, 1))
draw_box(x, time, "quarter", legend.xy = c(0, 1))
draw_box(x, time, "month",
  legend.orientation = "h",
  legend.xy = c(0, 1),
  legend.yanchor = "bottom"
)
# (Note how the boxplots widen when the period includes data from both dat1 and dat2)
```

draw_calibration	<i>Draw calibration plot</i>
------------------	------------------------------

Description

Draw calibration plot

Usage

```
draw_calibration(
  true_labels,
  predicted_prob,
  n_bins = 10L,
  bin_method = c("quantile", "equidistant"),
  binclasspos = 2L,
  main = NULL,
  subtitle = NULL,
  xlab = "Mean predicted probability",
  ylab = "Empirical risk",
```

```

    show_marginal_x = TRUE,
    marginal_x_y = -0.02,
    marginal_col = NULL,
    marginal_size = 10,
    mode = "markers+lines",
    show_brier = TRUE,
    theme = choose_theme(getOption("rtemis_theme")),
    filename = NULL,
    ...
)

```

Arguments

<code>true_labels</code>	Factor or list of factors with true class labels
<code>predicted_prob</code>	Numeric vector or list of numeric vectors with predicted probabilities
<code>n_bins</code>	Integer: Number of windows to split the data into
<code>bin_method</code>	Character: "quantile" or "equidistant": Method to bin the estimated probabilities.
<code>binclasspos</code>	Integer: Index of the positive class. The convention used in the package is the second level is the positive class.
<code>main</code>	Character: Main title
<code>subtitle</code>	Character: Subtitle, placed bottom right of plot
<code>xlab</code>	Character: x-axis label
<code>ylab</code>	Character: y-axis label
<code>show_marginal_x</code>	Logical: Add marginal plot of distribution of estimated probabilities
<code>marginal_x_y</code>	Numeric: y position of marginal plot
<code>marginal_col</code>	Character: Color of marginal plot
<code>marginal_size</code>	Numeric: Size of marginal plot
<code>mode</code>	Character: "lines", "markers", "lines+markers": How to plot.
<code>show_brier</code>	Logical: If TRUE, add Brier scores to trace names.
<code>theme</code>	Theme object.
<code>filename</code>	Character: Path to save output.
<code>...</code>	Additional arguments passed to draw_scatter

Value

plotly object.

Author(s)

EDG

Examples

```
# Synthetic data with n cases
n <- 500L
true_labels <- factor(sample(c("A", "B"), n, replace = TRUE))
# Synthetic probabilities where A has mean 0.25 and B has mean 0.75
predicted_prob <- ifelse(true_labels == "A",
  rbeta(n, 2, 6),
  rbeta(n, 6, 2)
)
draw_calibration(true_labels, predicted_prob)
```

draw_confusion	<i>Plot confusion matrix</i>
----------------	------------------------------

Description

Plot confusion matrix

Usage

```
draw_confusion(
  x,
  xlab = "Predicted",
  ylab = "Reference",
  true_col = "#43A4AC",
  false_col = "#FA9860",
  font_size = 18,
  main = NULL,
  main_y = 1,
  main_yanchor = "bottom",
  theme = choose_theme(getOption("rtemis_theme")),
  margin = list(l = 20, r = 5, b = 5, t = 20),
  filename = NULL,
  file_width = 500,
  file_height = 500,
  file_scale = 1
)
```

Arguments

x	ClassificationMetrics object produced by <code>rtemis::classification_metrics()</code> or confusion matrix where rows are the reference and columns are the estimated classes. For binary classification, the first row and column are the positive class.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
true_col	Color for true positives & true negatives.

false_col	Color for false positives & false negatives.
font_size	Integer: font size.
main	Character: plot title.
main_y	Numeric: y position of the title.
main_ynanchor	Character: y anchor of the title.
theme	Theme object.
margin	List: Plot margins.
filename	Character: file name to save the plot.
file_width	Numeric: width of the file.
file_height	Numeric: height of the file.
file_scale	Numeric: scale of the file.

Value

plotly object.

Author(s)

EDG

Examples

```
# Assume positive class is "b"
true_labels <- factor(c("a", "a", "a", "b", "b", "b", "b", "b", "b", "b"))
predicted_labels <- factor(c("a", "b", "a", "b", "b", "a", "b", "b", "b", "a"))
predicted_prob <- c(0.3, 0.55, 0.45, 0.75, 0.57, 0.3, 0.8, 0.63, 0.62, 0.39)
metrics <- classification_metrics(true_labels, predicted_labels, predicted_prob)
draw_confusion(metrics)
```

draw_dist

Draw Distributions using Histograms and Density Plots

Description

Draw Distributions using Histograms and Density Plots using plotly.

Usage

```
draw_dist(
  x,
  type = c("density", "histogram"),
  mode = c("overlap", "ridge"),
  group = NULL,
  main = NULL,
```

```

xlab = NULL,
ylab = NULL,
col = NULL,
alpha = 0.75,
plot_bg = NULL,
theme = choose_theme(getOption("rtemis_theme")),
palette = getOption("rtemis_palette", "rtms"),
axes_square = FALSE,
group_names = NULL,
font_size = 16,
font_alpha = 0.8,
legend = NULL,
legend_xy = c(0, 1),
legend_col = NULL,
legend_bg = "#FFFFFF00",
legend_border_col = "#FFFFFF00",
bargap = 0.05,
vline = NULL,
vline_col = theme[["fg"]],
vline_width = 1,
vline_dash = "dot",
text = NULL,
text_x = 1,
text_xref = "paper",
text_xanchor = "left",
text_y = 1,
text_yref = "paper",
text_yanchor = "top",
text_col = theme[["fg"]],
margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
automargin_x = TRUE,
automargin_y = TRUE,
zerolines = FALSE,
density_kernel = "gaussian",
density_bw = "SJ",
histnorm = c("", "density", "percent", "probability", "probability density"),
histfunc = c("count", "sum", "avg", "min", "max"),
hist_n_bins = 20,
barmode = "overlay",
ridge_sharex = TRUE,
ridge_y_labs = FALSE,
ridge_order_on_mean = TRUE,
displayModeBar = TRUE,
modeBar_file_format = "svg",
width = NULL,
height = NULL,
filename = NULL,
file_width = 500,

```

```

    file_height = 500,
    file_scale = 1
)

```

Arguments

x	Numeric vector / data.frame / list: Input. If not a vector, each column / each element is drawn.
type	Character: "density" or "histogram".
mode	Character: "overlap", "ridge". How to plot different groups; on the same axes ("overlap"), or on separate plots with the same x-axis ("ridge").
group	Vector: Will be converted to factor; levels define group members.
main	Character: Main title for the plot.
xlab	Character: Label for the x-axis.
ylab	Character: Label for the y-axis.
col	Color: Colors for the plot.
alpha	Numeric: Alpha transparency for plot elements.
plot_bg	Color: Background color for plot area.
theme	Theme object.
palette	Character: Color palette to use.
axes_square	Logical: If TRUE, draw a square plot to fill the graphic device.
group_names	Character: Names for the groups.
font_size	Numeric: Font size for plot text.
font_alpha	Numeric: Alpha transparency for font.
legend	Logical: If TRUE, draw legend. If NULL, set to TRUE if x is a list of more than 1 element.
legend_xy	Numeric, vector, length 2: Relative x, y position for legend.
legend_col	Color: Color for the legend text.
legend_bg	Color: Background color for legend.
legend_border_col	Color: Border color for legend.
bargap	Numeric: The gap between adjacent histogram bars in plot fraction.
vline	Numeric, vector: If defined, draw a vertical line at this x value(s).
vline_col	Color: Color for vline.
vline_width	Numeric: Width for vline.
vline_dash	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot".
text	Character: If defined, add this text over the plot.
text_x	Numeric: x-coordinate for text.
text_xref	Character: "x": text_x refers to plot's x-axis; "paper": text_x refers to plotting area from 0-1.

text_xanchor	Character: "auto", "left", "center", "right".
text_y	Numeric: y-coordinate for text.
text_yref	Character: "y": text_y refers to plot's y-axis; "paper": text_y refers to plotting area from 0-1.
text_yanchor	Character: "auto", "top", "middle", "bottom".
text_col	Color: Color for text.
margin	List: Margins for the plot.
automargin_x	Logical: If TRUE, automatically adjust x-axis margins.
automargin_y	Logical: If TRUE, automatically adjust y-axis margins.
zerolines	Logical: If TRUE, draw lines at $y = 0$.
density_kernel	Character: Kernel to use for density estimation.
density_bw	Character: Bandwidth to use for density estimation.
histnorm	Character: "", "density", "percent", "probability", "probability density".
histfunc	Character: "count", "sum", "avg", "min", "max".
hist_n_bins	Integer: Number of bins to use if type = "histogram".
barmode	Character: Barmode for histogram. One of "overlay", "stack", "relative", "group".
ridge_sharex	Logical: If TRUE, draw single x-axis when mode = "ridge".
ridge_y_labs	Logical: If TRUE, show individual y labels when mode = "ridge".
ridge_order_on_mean	Logical: If TRUE, order groups by mean value when mode = "ridge".
displayModeBar	Logical: If TRUE, display the mode bar.
modeBar_file_format	Character: File format for mode bar.
width	Numeric: Force plot size to this width. If NULL, fill available space.
height	Numeric: Force plot size to this height. If NULL, fill available space.
filename	Character: Path to file to save static plot.
file_width	Integer: File width in pixels for when filename is set.
file_height	Integer: File height in pixels for when filename is set.
file_scale	Numeric: If saving to file, scale plot by this number.

Details

See docs.rtemis.org/r for detailed documentation.

If input is data.frame, non-numeric variables will be removed.

Value

plotly object.

Author(s)

EDG

Examples

```
# Will automatically use only numeric columns
draw_dist(iris)
draw_dist(iris[["Sepal.Length"]], group = iris[["Species"]])
```

draw_fit

*True vs. Predicted Plot***Description**

A draw_scatter wrapper for plotting true vs. predicted values

Usage

```
draw_fit(
  x,
  y,
  xlab = "True",
  ylab = "Predicted",
  fit = "glm",
  se_fit = TRUE,
  axes_square = TRUE,
  axes_equal = TRUE,
  diagonal = TRUE,
  ...
)
```

Arguments

x	Numeric, vector/data.frame/list: True values. If y is NULL and NCOL(x) > 1, first two columns used as x and y, respectively
y	Numeric, vector/data.frame/list: Predicted values
xlab	Character: x-axis label.
ylab	Character: y-axis label.
fit	Character: Fit method.
se_fit	Logical: If TRUE, include standard error of the fit.
axes_square	Logical: If TRUE, draw a square plot.
axes_equal	Logical: If TRUE, set equal scaling for axes.
diagonal	Logical: If TRUE, add diagonal line.
...	Additional arguments passed to draw_scatter

Value

plotly object.

Author(s)

EDG

Examples

```
x <- rnorm(500)
y <- x + rnorm(500)
draw_fit(x, y)
```

draw_graphD3

*Plot graph using **networkD3*****Description**Plot graph using **networkD3****Usage**

```
draw_graphD3(
  net,
  groups = NULL,
  color_scale = NULL,
  edge_col = NULL,
  node_col = NULL,
  node_alpha = 0.5,
  edge_alpha = 0.33,
  zoom = TRUE,
  legend = FALSE,
  palette = get_palette(getOption("rtemis_palette")),
  theme = choose_theme(getOption("rtemis_theme")),
  ...
)
```

Arguments

net	igraph network.
groups	Vector, length n nodes indicating group/cluster/community membership of nodes in net.
color_scale	D3 colorscale (e.g. <code>networkD3::JS("d3.scaleOrdinal(d3.schemeCategory20b);")</code>).
edge_col	Color for edges.
node_col	Color for nodes.
node_alpha	Float [0, 1]: Node opacity.
edge_alpha	Float [0, 1]: Edge opacity.
zoom	Logical: If TRUE, graph is zoomable.

legend	Logical: If TRUE, display legend for groups.
palette	Character vector: Colors to use.
theme	Theme object.
...	Additional arguments to pass to networkD3.

Value

forceNetwork object.

Author(s)

EDG

Examples

```
library(igraph)
g <- make_ring(10)
draw_graphD3(g)
```

draw_graphjs

*Plot network using **threejs::graphjs***

Description

Interactive plotting of an **igraph** net using **threejs**.

Usage

```
draw_graphjs(
  net,
  vertex_size = 1,
  vertex_col = NULL,
  vertex_label_col = NULL,
  vertex_label_alpha = 0.66,
  vertex_frame_col = NA,
  vertex_label = NULL,
  vertex_shape = "circle",
  edge_col = NULL,
  edge_alpha = 0.5,
  edge_curved = 0.35,
  edge_width = 2,
  layout = c("fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  coords = NULL,
  layout_args = list(),
  cluster = NULL,
  groups = NULL,
```

```

    cluster_config = list(),
    cluster_mark_groups = TRUE,
    cluster_color_vertices = FALSE,
    main = "",
    theme = choose_theme(getOption("rtemis_theme")),
    palette = getOption("rtemis_palette", "rtms"),
    filename = NULL,
    ...
)

```

Arguments

<code>net</code>	igraph network.
<code>vertex_size</code>	Numeric: Vertex size.
<code>vertex_col</code>	Color for vertices.
<code>vertex_label_col</code>	Color for vertex labels.
<code>vertex_label_alpha</code>	Numeric: Transparency for <code>vertex_label_col</code> .
<code>vertex_frame_col</code>	Color for vertex border (frame).
<code>vertex_label</code>	Character vector: Vertex labels. If NULL, keep existing names in <code>net</code> if any. Set to NA to avoid printing vertex labels.
<code>vertex_shape</code>	Character, vector, length 1 or N nodes: Vertex shape. See <code>graphjs("vertex_shape")</code> .
<code>edge_col</code>	Color for edges.
<code>edge_alpha</code>	Numeric: Transparency for edges.
<code>edge_curved</code>	Numeric: Curvature of edges.
<code>edge_width</code>	Numeric: Edge thickness.
<code>layout</code>	Character: one of: "fr", "dh", "drl", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama", corresponding to all the available layouts in igraph .
<code>coords</code>	Output of precomputed igraph layout. If provided, layout is ignored.
<code>layout_args</code>	List of arguments to pass to layout function.
<code>cluster</code>	Character: one of: "edge_betweenness", "fast_greedy", "infomap", "label_prop", "leading_eigen", "louvain", "optimal", "spinglass", "walktrap", corresponding to all the available igraph clustering functions.
<code>groups</code>	Output of precomputed igraph clustering. If provided, cluster is ignored.
<code>cluster_config</code>	List of arguments to pass to cluster function.
<code>cluster_mark_groups</code>	Logical: If TRUE, draw polygons to indicate clusters, if groups or cluster are defined.
<code>cluster_color_vertices</code>	Logical: If TRUE, color vertices by cluster membership.
<code>main</code>	Character: Main title.

theme	Theme object.
palette	Color vector or name of rtemis palette.
filename	Character: If provided, save plot to this filepath.
...	Extra arguments to pass to threejs::graphjs().

Value

scatterplotThree object.

Author(s)

EDG

Examples

```
library(igraph)
g <- make_ring(10)
draw_graphjs(g)
```

draw_heatmap

Interactive Heatmaps

Description

Draw interactive heatmaps using heatmaply.

Usage

```
draw_heatmap(
  x,
  Rowv = TRUE,
  Colv = TRUE,
  cluster = FALSE,
  symm = FALSE,
  cellnote = NULL,
  colorgrad_n = 101,
  colors = NULL,
  space = "rgb",
  lo = "#18A3AC",
  lomid = NULL,
  mid = NULL,
  midhi = NULL,
  hi = "#F48024",
  k_row = 1,
  k_col = 1,
  grid_gap = 0,
```

```

    limits = NULL,
    margins = NULL,
    main = NULL,
    xlab = NULL,
    ylab = NULL,
    key_title = NULL,
    showticklabels = NULL,
    colorbar_len = 0.7,
    plot_method = "plotly",
    theme = choose_theme(getOption("rtemis_theme")),
    row_side_colors = NULL,
    row_side_palette = NULL,
    col_side_colors = NULL,
    col_side_palette = NULL,
    font_size = NULL,
    padding = 0,
    displayModeBar = TRUE,
    modeBar_file_format = "svg",
    filename = NULL,
    file_width = 500,
    file_height = 500,
    file_scale = 1,
    ...
)

```

Arguments

<code>x</code>	Input matrix.
<code>Rowv</code>	Logical or dendrogram. If Logical: Compute dendrogram and reorder rows. If dendrogram: use as is, without reordering. See more at <code>heatmaply::heatmaply("Rowv")</code> .
<code>Colv</code>	Logical or dendrogram. If Logical: Compute dendrogram and reorder columns. If dendrogram: use as is, without reordering. See more at <code>heatmaply::heatmaply("Colv")</code> .
<code>cluster</code>	Logical: If TRUE, set <code>Rowv</code> and <code>Colv</code> to TRUE.
<code>symm</code>	Logical: If TRUE, treat <code>x</code> symmetrically - <code>x</code> must be a square matrix.
<code>cellnote</code>	Matrix with values to be displayed on hover. Set to "values" to display <code>ddSci(x)</code> .
<code>colorgrad_n</code>	Integer: Number of colors in gradient.
<code>colors</code>	Character vector: Colors to use in gradient.
<code>space</code>	Character: Color space to use.
<code>lo</code>	Character: Color for low values.
<code>lomid</code>	Character: Color for low-mid values.
<code>mid</code>	Character: Color for mid values.
<code>midhi</code>	Character: Color for mid-high values.
<code>hi</code>	Character: Color for high values.
<code>k_row</code>	Integer: Number of desired number of groups by which to color dendrogram branches in the rows.

k_col	Integer: Number of desired number of groups by which to color dendrogram branches in the columns.
grid_gap	Integer: Space between cells.
limits	Float, length 2: Determine color range. If NULL, automatically centers values around 0.
margins	Float, length 4: Heatmap margins.
main	Character: Main title.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
key_title	Character: Title for the color key.
showticklabels	Logical: If TRUE, show tick labels.
colorbar_len	Numeric: Length of the colorbar.
plot_method	Character: Plot method to use.
theme	Theme object.
row_side_colors	Data frame: Column names will be label names, cells should be label colors. See <code>heatmaply::heatmaply("row_side_colors")</code> .
row_side_palette	Color palette function. See <code>heatmaply::heatmaply("row_side_palette")</code> .
col_side_colors	Data frame: Column names will be label names, cells should be label colors. See <code>heatmaply::heatmaply("col_side_colors")</code> .
col_side_palette	Color palette function. See <code>heatmaply::heatmaply("col_side_palette")</code> .
font_size	Numeric: Font size.
padding	Numeric: Padding around the plot, in pixels.
displayModeBar	Logical: If TRUE, display the plotly mode bar.
modeBar_file_format	Character: File format for image exports from the mode bar.
filename	Character: File name to save the plot.
file_width	Numeric: Width of exported image.
file_height	Numeric: Height of exported image.
file_scale	Numeric: Scale of exported image.
...	Additional arguments to be passed to <code>heatmaply::heatmaply</code> .

Details

See docs.rtemis.org/r for detailed documentation. 'heatmaply' unfortunately forces loading of the 'colorspace' namespace.

Value

plotly object.

Author(s)

EDG

Examples

```
x <- rnormmat(200, 20)
xcor <- cor(x)
draw_heatmap(xcor)
```

draw_leaflet

*Plot interactive choropleth map using **leaflet*****Description**Plot interactive choropleth map using **leaflet****Usage**

```
draw_leaflet(
  fips,
  values,
  names = NULL,
  fillOpacity = 1,
  color_mapping = c("Numeric", "Bin"),
  col_lo = "#0290EE",
  col_hi = "#FE4AA3",
  col_na = "#303030",
  col_highlight = "#FE8A4F",
  col_interpolate = c("linear", "spline"),
  col_bins = 21,
  domain = NULL,
  weight = 0.5,
  color = "black",
  alpha = 1,
  bg_tile_provider = leaflet::providers[["CartoDB.Positron"]],
  bg_tile_alpha = 0.67,
  fg_tile_provider = leaflet::providers[["CartoDB.PositronOnlyLabels"]],
  legend_position = c("topright", "bottomright", "bottomleft", "topleft"),
  legend_alpha = 0.8,
  legend_title = NULL,
  init_lng = -98.5418083333333,
  init_lat = 39.2074138888889,
  init_zoom = 3,
  stroke = TRUE
)
```

Arguments

fips	Character vector: FIPS codes. (If numeric, it will be appropriately zero-padded).
values	Values to map to fips.
names	Character vector: Optional county names to appear on hover along values.
fillOpacity	Float: Opacity for fill colors.
color_mapping	Character: "Numeric" or "Bin".
col_lo	Overlay color mapped to lowest value.
col_hi	Overlay color mapped to highest value.
col_na	Color mapped to NA values.
col_highlight	Hover border color.
col_interpolate	Character: "linear" or "spline".
col_bins	Integer: Number of color bins to create if color_mapping = "Bin".
domain	Limits for mapping colors to values. If NULL, set to the range of values.
weight	Float: Weight of county border lines.
color	Color of county border lines.
alpha	Float: Overlay transparency.
bg_tile_provider	Background tile (below overlay colors), one of leaflet::providers.
bg_tile_alpha	Float: Background tile transparency.
fg_tile_provider	Foreground tile (above overlay colors), one of leaflet::providers.
legend_position	Character: One of: "topright", "bottomright", "bottomleft", "topleft".
legend_alpha	Float: Legend box transparency.
legend_title	Character: Legend title. If NULL, set to the name of the values variable.
init_lng	Float: Longitude (in decimal form) to center the map around.
init_lat	Float: Latitude (in decimal form) to center the map around.
init_zoom	Integer: Initial zoom level (depends on device, i.e. window, size).
stroke	Logical: If TRUE, draw polygon borders.

Value

leaflet object.

Author(s)

EDG

Examples

```
fips <- c(06075, 42101)
population <- c(874961, 1579000)
names <- c("SF", "Philly")
draw_leaflet(fips, population, names)
```

draw_pie

*Interactive Pie Chart***Description**

Draw interactive pie charts using plotly.

Usage

```
draw_pie(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  alpha = 0.8,
  bg = NULL,
  plot_bg = NULL,
  theme = choose_theme(getOption("rtemis_theme")),
  palette = get_palette(getOption("rtemis_palette")),
  category_names = NULL,
  textinfo = "label+percent",
  font_size = 16,
  labs_col = NULL,
  legend = TRUE,
  legend_col = NULL,
  sep_col = NULL,
  margin = list(b = 50, l = 50, t = 50, r = 20),
  padding = 0,
  displayModeBar = TRUE,
  modeBar_file_format = "svg",
  filename = NULL,
  file_width = 500,
  file_height = 500,
  file_scale = 1
)
```

Arguments

x data.frame: Input: Either a) 1 numeric column with categories defined by row-names, or b) two columns, the first is category names, the second numeric or c) a numeric vector with categories defined using the `category.names` argument.

main	Character: Plot title. If NULL, set to colnames(x)[1].
xlab	Character: x-axis label.
ylab	Character: y-axis label.
alpha	Numeric: Alpha for the pie slices.
bg	Character: Background color.
plot_bg	Character: Plot background color.
theme	Theme object.
palette	Character vector: Colors to use.
category_names	Character, vector, length = NROW(x): Category names. If NULL, uses either rownames(x), or the first column of x if ncol(x) = 2.
textinfo	Character: Info to show over each slice: "label", "percent", "label+percent".
font_size	Integer: Font size for labels.
labs_col	Character: Color of labels.
legend	Logical: If TRUE, show legend.
legend_col	Character: Color for legend.
sep_col	Character: Separator color.
margin	List: Margin settings.
padding	Numeric: Padding around the plot, in pixels.
displayModeBar	Logical: If TRUE, display the plotly mode bar.
modeBar_file_format	Character: File format for image exports from the mode bar.
filename	Character: File name to save plot.
file_width	Integer: Width for saved file.
file_height	Integer: Height for saved file.
file_scale	Numeric: Scale for saved file.

Value

plotly object.

Author(s)

EDG

Examples

```
draw_pie(VADeaths[, 1, drop = FALSE])
```

draw_protein	<i>Plot an amino acid sequence with annotations</i>
--------------	---

Description

Plot an amino acid sequence with multiple site and/or region annotations.

Usage

```
draw_protein(  
  x,  
  site = NULL,  
  region = NULL,  
  ptm = NULL,  
  cleavage_site = NULL,  
  variant = NULL,  
  disease_variants = NULL,  
  n_per_row = NULL,  
  main = NULL,  
  main_xy = c(0.055, 0.975),  
  main_xref = "paper",  
  main_yref = "paper",  
  main_xanchor = "middle",  
  main_yanchor = "top",  
  layout = c("simple", "grid", "1curve", "2curve"),  
  show_markers = TRUE,  
  show_labels = TRUE,  
  font_size = 18,  
  label_col = NULL,  
  scatter_mode = "markers+lines",  
  marker_size = 28,  
  marker_col = NULL,  
  marker_alpha = 1,  
  marker_symbol = "circle",  
  line_col = NULL,  
  line_alpha = 1,  
  line_width = 2,  
  show_full_names = TRUE,  
  region_scatter_mode = "markers+lines",  
  region_style = 3,  
  region_marker_size = marker_size,  
  region_marker_alpha = 0.6,  
  region_marker_symbol = "circle",  
  region_line_dash = "solid",  
  region_line_shape = "line",  
  region_line_smoothing = 1,  
  region_line_width = 1,
```

```

region_line_alpha = 0.6,
theme = choose_theme(getOption("rtemis_theme")),
region_palette = getOption("rtemis_palette", "rtms"),
region_outline_only = FALSE,
region_outline_pad = 2,
region_pad = 0.35,
region_fill_alpha = 0.1666666,
region_fill_shape = "line",
region_fill_smoothing = 1,
bpadcx = 0.5,
bpadcy = 0.5,
site_marker_size = marker_size,
site_marker_symbol = marker_symbol,
site_marker_alpha = 1,
site_border_width = 1.5,
site_palette = getOption("rtemis_palette", "rtms"),
variant_col = "#FA6E1E",
disease_variant_col = "#E266AE",
showlegend_ptm = TRUE,
ptm_col = NULL,
ptm_symbol = "circle",
ptm_offset = 0.12,
ptm_pad = 0.35,
ptm_marker_size = marker_size/4.5,
clv_col = NULL,
clv_symbol = "triangle-down",
clv_offset = 0.12,
clv_pad = 0.35,
clv_marker_size = marker_size/4,
annotate_position_every = 10,
annotate_position_alpha = 0.5,
annotate_position_ay = -0.4 * marker_size,
position_font_size = font_size - 6,
legend_xy = c(0.97, 0.954),
legend_xanchor = "left",
legend_yanchor = "top",
legend_orientation = "v",
legend_col = NULL,
legend_bg = "#FFFFFF00",
legend_border_col = "#FFFFFF00",
legend_borderwidth = 0,
legend_group_gap = 0,
margin = list(b = 0, l = 0, t = 0, r = 0, pad = 0),
showgrid_x = FALSE,
showgrid_y = FALSE,
automargin_x = TRUE,
automargin_y = TRUE,
xaxis_aurorange = TRUE,

```

```

    yaxis_aurange = "reversed",
    scaleanchor_y = "x",
    scaleratio_y = 1,
    hoverlabel_align = "left",
    displayModeBar = TRUE,
    modeBar_file_format = "svg",
    scrollZoom = TRUE,
    filename = NULL,
    file_width = 1320,
    file_height = 990,
    file_scale = 1,
    width = NULL,
    height = NULL,
    verbosity = 1L
)

```

Arguments

x	Character vector: amino acid sequence (1-letter abbreviations) OR a3 object OR Character: path to JSON file OR Character: UniProt accession number.
site	Named list of lists with indices of sites. These will be highlighted by coloring the border of markers.
region	Named list of lists with indices of regions. These will be highlighted by coloring the markers and lines of regions using the palette colors.
ptm	List of post-translational modifications.
cleavage_site	List of cleavage sites.
variant	List of variant information.
disease_variants	List of disease variant information.
n_per_row	Integer: Number of amino acids to show per row.
main	Character: Main title.
main_xy	Numeric vector, length 2: x and y coordinates for title. e.g. if main_xref and main_yref are "paper": c(0.055, .975) is top left, c(.5, .975) is top and middle.
main_xref	Character: xref for title.
main_yref	Character: yref for title.
main_xanchor	Character: xanchor for title.
main_yanchor	Character: yanchor for title.
layout	Character: "simple", "grid", "1curve", "2curve": type of layout to use.
show_markers	Logical: If TRUE, show amino acid markers.
show_labels	Logical: If TRUE, annotate amino acids with elements.
font_size	Integer: Font size for labels.
label_col	Color for labels.

<code>scatter_mode</code>	Character: Mode for scatter plot.
<code>marker_size</code>	Integer: Size of markers.
<code>marker_col</code>	Color for markers.
<code>marker_alpha</code>	Numeric: Alpha for markers.
<code>marker_symbol</code>	Character: Symbol for markers.
<code>line_col</code>	Color for lines.
<code>line_alpha</code>	Numeric: Alpha for lines.
<code>line_width</code>	Numeric: Width for lines.
<code>show_full_names</code>	Logical: If TRUE, show full names of amino acids.
<code>region_scatter_mode</code>	Character: Mode for scatter plot.
<code>region_style</code>	Integer: Style for regions.
<code>region_marker_size</code>	Integer: Size of region markers.
<code>region_marker_alpha</code>	Numeric: Alpha for region markers.
<code>region_marker_symbol</code>	Character: Symbol for region markers.
<code>region_line_dash</code>	Character: Dash for region lines.
<code>region_line_shape</code>	Character: Shape for region lines.
<code>region_line_smoothing</code>	Numeric: Smoothing for region lines.
<code>region_line_width</code>	Numeric: Width for region lines.
<code>region_line_alpha</code>	Numeric: Alpha for region lines.
<code>theme</code>	Theme object.
<code>region_palette</code>	Named list of colors for regions.
<code>region_outline_only</code>	Logical: If TRUE, only show outline of regions.
<code>region_outline_pad</code>	Numeric: Padding for region outline.
<code>region_pad</code>	Numeric: Padding for region.
<code>region_fill_alpha</code>	Numeric: Alpha for region fill.
<code>region_fill_shape</code>	Character: Shape for region fill.
<code>region_fill_smoothing</code>	Numeric: Smoothing for region fill.

bpadcx	Numeric: Padding for region border.
bpadcy	Numeric: Padding for region border.
site_marker_size	Integer: Size of site markers.
site_marker_symbol	Character: Symbol for site markers.
site_marker_alpha	Numeric: Alpha for site markers.
site_border_width	Numeric: Width for site borders.
site_palette	Named list of colors for sites.
variant_col	Color for variants.
disease_variant_col	Color for disease variants.
showlegend_ptm	Logical: If TRUE, show legend for PTMs.
ptm_col	Named list of colors for PTMs.
ptm_symbol	Character: Symbol for PTMs.
ptm_offset	Numeric: Offset for PTMs.
ptm_pad	Numeric: Padding for PTMs.
ptm_marker_size	Integer: Size of PTM markers.
clv_col	Color for cleavage site annotations.
clv_symbol	Character: Symbol for cleavage site annotations.
clv_offset	Numeric: Offset for cleavage site annotations.
clv_pad	Numeric: Padding for cleavage site annotations.
clv_marker_size	Integer: Size of cleavage site annotation markers.
annotate_position_every	Integer: Annotate every nth position.
annotate_position_alpha	Numeric: Alpha for position annotations.
annotate_position_ay	Numeric: Y offset for position annotations.
position_font_size	Integer: Font size for position annotations.
legend_xy	Numeric vector, length 2: x and y coordinates for legend.
legend_xanchor	Character: xanchor for legend.
legend_yanchor	Character: yanchor for legend.
legend_orientation	Character: Orientation for legend.
legend_col	Color for legend.

legend_bg	Color for legend background.
legend_border_col	Color for legend border.
legend_borderwidth	Numeric: Width for legend border.
legend_group_gap	Numeric: Gap between legend groups.
margin	List: Margin settings.
showgrid_x	Logical: If TRUE, show x grid.
showgrid_y	Logical: If TRUE, show y grid.
automargin_x	Logical: If TRUE, use automatic margin for x axis.
automargin_y	Logical: If TRUE, use automatic margin for y axis.
xaxis_autorange	Logical: If TRUE, use automatic range for x axis.
yaxis_autorange	Logical: If TRUE, use automatic range for y axis.
scaleanchor_y	Character: Scale anchor for y axis.
scaleratio_y	Numeric: Scale ratio for y axis.
hoverlabel_align	Character: Alignment for hover label.
displayModeBar	Logical: If TRUE, display mode bar.
modeBar_file_format	Character: File format for mode bar.
scrollZoom	Logical: If TRUE, enable scroll zoom.
filename	Character: File name to save plot.
file_width	Integer: Width for saved file.
file_height	Integer: Height for saved file.
file_scale	Numeric: Scale for saved file.
width	Integer: Width for plot.
height	Integer: Height for plot.
verbosity	Integer: Verbosity level.

Value

plotly object.

Author(s)

EDG

Examples

```
## Not run:
# Reads sequence from UniProt server
tau <- seqinr::read.fasta("https://rest.uniprot.org/uniprotkb/P10636.fasta",
  seqtype = "AA"
)
draw_protein(as.character(tau[[1]]))

# or directly using the UniProt accession number:
draw_protein("P10636")

## End(Not run)
```

draw_pvals	<i>Barplot p-values using draw_bar</i>
------------	--

Description

Plot 1 - p-values as a barplot

Usage

```
draw_pvals(
  x,
  xnames = NULL,
  yname = NULL,
  p_adjust_method = "none",
  pval_hline = 0.05,
  hline_col = rtemis_colors[["red"]],
  hline_dash = "dash",
  ...
)
```

Arguments

x	Float, vector: p-values.
xnames	Character, vector: feature names.
yname	Character: outcome name.
p_adjust_method	Character: method for p.adjust .
pval_hline	Float: Significance level at which to plot horizontal line.
hline_col	Color for pval_hline.
hline_dash	Character: type of line to draw.
...	Additional arguments passed to draw_bar .

Value

plotly object.

Author(s)

EDG

Examples

```
draw_pvals(c(0.01, 0.02, 0.03), xnames = c("Feature1", "Feature2", "Feature3"))
```

draw_roc	<i>Draw ROC curve</i>
----------	-----------------------

Description

Draw ROC curve

Usage

```
draw_roc(  
  true_labels,  
  predicted_prob,  
  main = NULL,  
  theme = choose_theme(getOption("rtemis_theme")),  
  palette = get_palette(getOption("rtemis_palette")),  
  legend = TRUE,  
  legend_title = "Group (AUC)",  
  legend_xy = c(1, 0),  
  legend_xanchor = "right",  
  legend_yanchor = "bottom",  
  auc_dp = 3L,  
  xlim = c(-0.05, 1.05),  
  ylim = c(-0.05, 1.05),  
  diagonal = TRUE,  
  diagonal_col = NULL,  
  axes_square = TRUE,  
  filename = NULL,  
  ...  
)
```

Arguments

true_labels Factor: True outcome labels.

predicted_prob	Numeric vector [0, 1]: Predicted probabilities for the positive class (i.e. second level of outcome). Or, for multiclass, a matrix of predicted probabilities with one column per class. Or, a list of such vectors/matrices to draw multiple ROC curves on the same plot.
main	Character: Main title for the plot.
theme	Theme object.
palette	Character vector: Colors to use.
legend	Logical: If TRUE, draw legend.
legend_title	Character: Title for the legend.
legend_xy	Numeric vector: Position of the legend in the form c(x, y).
legend_xanchor	Character: X anchor for the legend.
legend_yanchor	Character: Y anchor for the legend.
auc_dp	Integer: Number of decimal places for AUC values.
xlim	Numeric vector: Limits for the x-axis.
ylim	Numeric vector: Limits for the y-axis.
diagonal	Logical: If TRUE, draw diagonal line.
diagonal_col	Character: Color for the diagonal line.
axes_square	Logical: If TRUE, make axes square.
filename	Character: If provided, save the plot to this file.
...	Additional arguments passed to draw_scatter .

Value

plotly object.

Author(s)

EDG

Examples

```
# Binary classification
true_labels <- factor(c("A", "B", "A", "A", "B", "A", "B", "B", "A", "B"))
predicted_prob <- c(0.1, 0.4, 0.35, 0.8, 0.65, 0.2, 0.9, 0.55, 0.3, 0.7)
draw_roc(true_labels, predicted_prob)
```

draw_scatter*Interactive Scatter Plots*

Description

Draw interactive scatter plots using plotly.

Usage

```
draw_scatter(  
  x,  
  y = NULL,  
  fit = NULL,  
  se_fit = FALSE,  
  se_times = 1.96,  
  include_fit_name = TRUE,  
  clustering_config = NULL,  
  group = NULL,  
  rsq = TRUE,  
  mode = "markers",  
  order_on_x = NULL,  
  main = NULL,  
  subtitle = NULL,  
  xlab = NULL,  
  ylab = NULL,  
  alpha = NULL,  
  theme = choose_theme(getOption("rtemis_theme")),  
  palette = get_palette(getOption("rtemis_palette")),  
  axes_square = FALSE,  
  group_names = NULL,  
  font_size = 16,  
  marker_col = NULL,  
  marker_size = 8,  
  symbol = "circle",  
  fit_col = NULL,  
  fit_alpha = 0.8,  
  fit_lwd = 2.5,  
  line_shape = "linear",  
  se_col = NULL,  
  se_alpha = 0.4,  
  scatter_type = "scatter",  
  show_marginal_x = FALSE,  
  show_marginal_y = FALSE,  
  marginal_x = x,  
  marginal_y = y,  
  marginal_x_y = NULL,  
  marginal_y_x = NULL,
```

```

marginal_col = NULL,
marginal_alpha = 0.333,
marginal_size = 10,
legend = NULL,
legend_title = NULL,
legend_trace = TRUE,
legend_xy = c(0, 0.98),
legend_xanchor = "left",
legend_yanchor = "auto",
legend_orientation = "v",
legend_col = NULL,
legend_bg = "#FFFFFF00",
legend_border_col = "#FFFFFF00",
legend_borderwidth = 0,
legend_group_gap = 0,
x_showspikes = FALSE,
y_showspikes = FALSE,
spikedash = "solid",
spikemode = "across",
spikesnap = "hovered data",
spikecolor = NULL,
spikethickness = 1,
margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
main_y = 1.01,
main_yanchor = "bottom",
subtitle_x = 0.02,
subtitle_y = 0.99,
subtitle_xref = "paper",
subtitle_yref = "paper",
subtitle_xanchor = "left",
subtitle_yanchor = "top",
automargin_x = TRUE,
automargin_y = TRUE,
xlim = NULL,
ylim = NULL,
axes_equal = FALSE,
diagonal = FALSE,
diagonal_col = NULL,
diagonal_dash = "dot",
diagonal_alpha = 0.66,
fit_params = NULL,
vline = NULL,
vline_col = theme[["fg"]],
vline_width = 1,
vline_dash = "dot",
hline = NULL,
hline_col = theme[["fg"]],
hline_width = 1,

```

```

hline_dash = "dot",
hovertext = NULL,
width = NULL,
height = NULL,
displayModeBar = TRUE,
modeBar_file_format = "svg",
scrollZoom = TRUE,
filename = NULL,
file_width = 500,
file_height = 500,
file_scale = 1,
verbosity = 0L
)

```

Arguments

x	Numeric, vector/data.frame/list: x-axis data. If y is NULL and NCOL(x) > 1, first two columns used as x and y, respectively.
y	Numeric, vector/data.frame/list: y-axis data.
fit	Character: Fit method.
se_fit	Logical: If TRUE, include standard error of the fit.
se_times	Numeric: Multiplier for standard error.
include_fit_name	Logical: If TRUE, include fit name in legend.
clustering_config	ClusteringConfig object: If provided, cluster on x and y and pass clusters to group.
group	Factor: Grouping variable.
rsq	Logical: If TRUE, print R-squared values in legend if fit is set.
mode	Character, vector: "markers", "lines", "markers+lines".
order_on_x	Logical: If TRUE, order x and y on x.
main	Character: Main title.
subtitle	Character: Subtitle.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
alpha	Numeric: Alpha for markers.
theme	Theme object.
palette	Character vector: Colors to use.
axes_square	Logical: If TRUE, draw a square plot.
group_names	Character: Names for groups.
font_size	Numeric: Font size.
marker_col	Color for markers.

marker_size	Numeric: Marker size.
symbol	Character: Marker symbol.
fit_col	Color for fit line.
fit_alpha	Numeric: Alpha for fit line.
fit_lwd	Numeric: Line width for fit line.
line_shape	Character: Line shape for line plots. Options: "linear", "hv", "vh", "hvh", "vhv".
se_col	Color for standard error band.
se_alpha	Numeric: Alpha for standard error band.
scatter_type	Character: Scatter plot type.
show_marginal_x	Logical: If TRUE, add marginal distribution line markers on x-axis.
show_marginal_y	Logical: If TRUE, add marginal distribution line markers on y-axis.
marginal_x	Numeric: Data for marginal distribution on x-axis.
marginal_y	Numeric: Data for marginal distribution on y-axis.
marginal_x_y	Numeric: Y position of marginal markers on x-axis.
marginal_y_x	Numeric: X position of marginal markers on y-axis.
marginal_col	Color for marginal markers.
marginal_alpha	Numeric: Alpha for marginal markers.
marginal_size	Numeric: Size of marginal markers.
legend	Logical: If TRUE, draw legend.
legend_title	Character: Title for legend.
legend_trace	Logical: If TRUE, draw legend trace. (For when you have fit and don't want a trace for the markers.)
legend_xy	Numeric: Position of legend.
legend_xanchor	Character: X anchor for legend.
legend_yanchor	Character: Y anchor for legend.
legend_orientation	Character: Orientation of legend.
legend_col	Color for legend text.
legend_bg	Color for legend background.
legend_border_col	Color for legend border.
legend_borderwidth	Numeric: Border width for legend.
legend_group_gap	Numeric: Gap between legend groups.
x_showspikes	Logical: If TRUE, show spikes on x-axis.
y_showspikes	Logical: If TRUE, show spikes on y-axis.

spikedash	Character: Dash type for spikes.
spikemode	Character: Spike mode.
spikesnap	Character: Spike snap mode.
spikecolor	Color for spikes.
spikethickness	Numeric: Thickness of spikes.
margin	List: Plot margins.
main_y	Numeric: Y position of main title.
main_ymargin	Character: Y anchor for main title.
subtitle_x	Numeric: X position of subtitle.
subtitle_y	Numeric: Y position of subtitle.
subtitle_xref	Character: X reference for subtitle.
subtitle_yref	Character: Y reference for subtitle.
subtitle_xmargin	Character: X anchor for subtitle.
subtitle_ymargin	Character: Y anchor for subtitle.
automargin_x	Logical: If TRUE, automatically adjust x-axis margins.
automargin_y	Logical: If TRUE, automatically adjust y-axis margins.
xlim	Numeric: Limits for x-axis.
ylim	Numeric: Limits for y-axis.
axes_equal	Logical: If TRUE, set equal scaling for axes.
diagonal	Logical: If TRUE, add diagonal line.
diagonal_col	Color for diagonal line.
diagonal_dash	Character {"solid", "dash", "dot", "dashdot", "longdash", "longdashdot"}: Dash type for diagonal line.
diagonal_alpha	Numeric: Alpha for diagonal line.
fit_params	Hyperparameters for fit.
vline	Numeric: X position for vertical line.
vline_col	Color for vertical line.
vline_width	Numeric: Width for vertical line.
vline_dash	Character: Dash type for vertical line.
hline	Numeric: Y position for horizontal line.
hline_col	Color for horizontal line.
hline_width	Numeric: Width for horizontal line.
hline_dash	Character: Dash type for horizontal line.
hovertext	List: Hover text for markers.
width	Numeric: Width of plot.
height	Numeric: Height of plot.

displayModeBar Logical: If TRUE, display mode bar.
modeBar_file_format Character: File format for mode bar.
scrollZoom Logical: If TRUE, enable scroll zoom.
filename Character: Filename to save plot.
file_width Numeric: Width of saved file.
file_height Numeric: Height of saved file.
file_scale Numeric: Scale of saved file.
verbosity Integer: Verbosity level.

Value

plotly object.

Author(s)

EDG

Examples

```
draw_spectrogram(iris[["Sepal.Length"]], iris[["Petal.Length"]],
  fit = "gam", se_fit = TRUE, group = iris[["Species"]]
)
```

draw_spectrogram	<i>Interactive Spectrogram</i>
------------------	--------------------------------

Description

Draw interactive spectrograms using plotly

Usage

```
draw_spectrogram(
  x,
  y,
  z,
  colorgrad_n = 101,
  colors = NULL,
  xlab = "Time",
  ylab = "Frequency",
  zlab = "Power",
  hover_xlab = xlab,
  hover_ylab = ylab,
  hover_zlab = zlab,
```

```

    zmin = NULL,
    zmax = NULL,
    zauto = TRUE,
    hoverlabel_align = "right",
    colorscale = "Jet",
    colorbar_y = 0.5,
    colorbar_yanchor = "middle",
    colorbar_xpad = 0,
    colorbar_ypad = 0,
    colorbar_len = 0.75,
    colorbar_title_side = "bottom",
    showgrid = FALSE,
    space = "rgb",
    lo = "#18A3AC",
    lomid = NULL,
    mid = NULL,
    midhi = NULL,
    hi = "#F48024",
    main = NULL,
    showticklabels = NULL,
    theme = choose_theme(getOption("rtemis_theme")),
    font_size = NULL,
    padding = 0,
    displayModeBar = TRUE,
    modeBar_file_format = "svg",
    filename = NULL,
    file_width = 500,
    file_height = 500,
    file_scale = 1
)

```

Arguments

<code>x</code>	Numeric: Time.
<code>y</code>	Numeric: Frequency.
<code>z</code>	Numeric: Power.
<code>colorgrad_n</code>	Integer: Number of colors in the gradient.
<code>colors</code>	Character: Custom colors for the gradient.
<code>xlab</code>	Character: x-axis label.
<code>ylab</code>	Character: y-axis label.
<code>zlab</code>	Character: z-axis label.
<code>hover_xlab</code>	Character: x-axis label for hover.
<code>hover_ylab</code>	Character: y-axis label for hover.
<code>hover_zlab</code>	Character: z-axis label for hover.
<code>zmin</code>	Numeric: Minimum value for color scale.

zmax	Numeric: Maximum value for color scale.
zauto	Logical: If TRUE, automatically set zmin and zmax.
hoverlabel_align	Character: Alignment of hover labels.
colorscale	Character: Color scale.
colorbar_y	Numeric: Y position of colorbar.
colorbar_ymanchor	Character: Y anchor of colorbar.
colorbar_xpad	Numeric: X padding of colorbar.
colorbar_ypad	Numeric: Y padding of colorbar.
colorbar_len	Numeric: Length of colorbar.
colorbar_title_side	Character: Side of colorbar title.
showgrid	Logical: If TRUE, show grid.
space	Character: Color space for gradient.
lo	Character: Low color for gradient.
lomid	Character: Low-mid color for gradient.
mid	Character: Mid color for gradient.
midhi	Character: Mid-high color for gradient.
hi	Character: High color for gradient.
main	Character: Main title.
showticklabels	Logical: If TRUE, show tick labels.
theme	Theme object.
font_size	Numeric: Font size.
padding	Numeric: Padding around the plot, in pixels.
displayModeBar	Logical: If TRUE, display the plotly mode bar.
modeBar_file_format	Character: File format for image exports from the mode bar.
filename	Character: Filename to save the plot.
file_width	Numeric: Width of exported image.
file_height	Numeric: Height of exported image.
file_scale	Numeric: Scale of exported image.

Details

To set custom colors, use a minimum of `lo` and `hi`, optionally also `lomid`, `mid`, `midhi` colors and set `colorscale = NULL`.

Value

plotly object.

Author(s)

EDG

Examples

```
# Example data
time <- seq(0, 10, length.out = 100)
freq <- seq(1, 100, length.out = 100)
power <- outer(time, freq, function(t, f) sin(t) * cos(f))
draw_spectrogram(
  x = time,
  y = freq,
  z = power
)
```

draw_survfit*Draw a survfit object*

Description

Draw a survfit object using [draw_scatter](#).

Usage

```
draw_survfit(
  x,
  mode = "lines",
  symbol = "cross",
  line_shape = "hv",
  xlim = NULL,
  ylim = NULL,
  xlab = "Time",
  ylab = "Survival",
  main = NULL,
  legend_xy = c(1, 1),
  legend_xanchor = "right",
  legend_yanchor = "top",
  theme = choose_theme(getOption("rtemis_theme")),
  nrisk_table = FALSE,
  filename = NULL,
  ...
)
```

Arguments

x	survfit object created by survival::survfit .
mode	Character, vector: "markers", "lines", "markers+lines".
symbol	Character: Symbol to use for the points.
line_shape	Character: Line shape for line plots. Options: "linear", "hv", "vh", "hvh", "vhv".
xlim	Numeric vector of length 2: x-axis limits.
ylim	Numeric vector of length 2: y-axis limits.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
main	Character: Main title.
legend_xy	Numeric: Position of legend.
legend_xanchor	Character: X anchor for legend.
legend_yanchor	Character: Y anchor for legend.
theme	Theme object.
nrisk_table	Logical: If TRUE, subplot a table of the number at risk at each time point.
filename	Character: Filename to save plot.
...	Additional arguments passed to draw_scatter .

Value

plotly object.

Author(s)

EDG

Examples

```
# Get the lung dataset
data(cancer, package = "survival")
sf1 <- survival::survfit(survival::Surv(time, status) ~ 1, data = lung)
draw_survfit(sf1)
sf2 <- survival::survfit(survival::Surv(time, status) ~ sex, data = lung)
draw_survfit(sf2)
# with N at risk table
draw_survfit(sf2)
```

draw_table

*Simple HTML table***Description**

Draw an html table using plotly

Usage

```
draw_table(
  x,
  .ddSci = TRUE,
  main = NULL,
  main_col = "black",
  main_x = 0,
  main_xanchor = "auto",
  fill_col = "#18A3AC",
  table_bg = "white",
  bg = "white",
  line_col = "white",
  lwd = 1,
  header_font_col = "white",
  table_font_col = "gray20",
  font_size = 14,
  font_family = "Helvetica Neue",
  margin = list(l = 0, r = 5, t = 30, b = 0, pad = 0)
)
```

Arguments

x	data.frame: Table to draw
.ddSci	Logical: If TRUE, apply ddSci to numeric columns.
main	Character: Table title.
main_col	Color: Title color.
main_x	Float [0, 1]: Align title: 0: left, .5: center, 1: right.
main_xanchor	Character: "auto", "left", "right": plotly's layout xanchor for title.
fill_col	Color: Used to fill header with column names and first column with row names.
table_bg	Color: Table background.
bg	Color: Background.
line_col	Color: Line color.
lwd	Float: Line width.
header_font_col	Color: Header font color.

table_font_col	Color: Table font color.
font_size	Integer: Font size.
font_family	Character: Font family.
margin	List: plotly's margins.

Value

plotly object.

Author(s)

EDG

Examples

```
df <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 35),
  Score = c(90.5, 85.0, 88.0)
)
p <- draw_table(
  df,
  main = "Sample Table",
  main_col = "#00b2b2"
)
```

draw_ts

Interactive Timeseries Plots

Description

Draw interactive timeseries plots using plotly

Usage

```
draw_ts(
  x,
  time,
  window = 7L,
  group = NULL,
  roll_fn = c("mean", "median", "max", "none"),
  roll_col = NULL,
  roll_alpha = 1,
  roll_lwd = 2,
  roll_name = NULL,
  alpha = NULL,
  align = "center",
```

```

group_names = NULL,
xlab = "Time",
n_xticks = 12,
scatter_type = "scatter",
legend = TRUE,
x_showspikes = TRUE,
y_showspikes = FALSE,
spikedash = "solid",
spikemode = "across",
spikesnap = "hovered data",
spikecolor = NULL,
spikethickness = 1,
displayModeBar = TRUE,
modeBar_file_format = "svg",
theme = choose_theme(getOption("rtemis_theme")),
palette = getOption("rtemis_palette", "rtms"),
filename = NULL,
file_width = 500,
file_height = 500,
file_scale = 1,
...
)

```

Arguments

x	Numeric vector of values to plot or list of vectors
time	Numeric or Date vector of time corresponding to values of x
window	Integer: apply roll_fn over this many units of time
group	Factor defining groups
roll_fn	Character: "mean", "median", "max", or "none": Function to apply on rolling windows of x. "none" disables the rolling line.
roll_col	Color for rolling line
roll_alpha	Numeric: transparency for rolling line
roll_lwd	Numeric: width of rolling line
roll_name	Rolling function name (for annotation)
alpha	Numeric [0, 1]: Transparency
align	Character: "center", "right", or "left"
group_names	Character vector of group names
xlab	Character: x-axis label
n_xticks	Integer: number of x-axis ticks to use (approximately)
scatter_type	Character: "scatter" or "lines"
legend	Logical: If TRUE, show legend
x_showspikes	Logical: If TRUE, show x-axis spikes on hover
y_showspikes	Logical: If TRUE, show y-axis spikes on hover

spikedash	Character: dash type string ("solid", "dot", "dash", "longdash", "dashdot", or "longdashdot") or a dash length list in px (eg "5px,10px,2px,2px")
spikemode	Character: If "toaxis", spike line is drawn from the data point to the axis the series is plotted on. If "across", the line is drawn across the entire plot area, and supercedes "toaxis". If "marker", then a marker dot is drawn on the axis the series is plotted on
spikesnap	Character: "data", "cursor", "hovered data". Determines whether spikelines are stuck to the cursor or to the closest datapoints.
spikecolor	Color for spike lines
spikethickness	Numeric: spike line thickness
displayModeBar	Logical: If TRUE, display plotly's modebar
modeBar_file_format	Character: modeBar image export file format
theme	Theme object.
palette	Character: palette name, or list of colors
filename	Character: Path to filename to save plot
file_width	Numeric: image export width
file_height	Numeric: image export height
file_scale	Numeric: image export scale
...	Additional arguments to be passed to draw_scatter

Value

plotly object.

Author(s)

EDG

Examples

```
time <- sample(seq(as.Date("2020-03-01"), as.Date("2020-09-23"), length.out = 140))
x1 <- rnorm(140)
x2 <- rnorm(140, 1, 1.2)
# Single timeseries
draw_ts(x1, time)
# Multiple timeseries input as list
draw_ts(list(Alpha = x1, Beta = x2), time)
# Multiple timeseries grouped by group, different lengths
time1 <- sample(seq(as.Date("2020-03-01"), as.Date("2020-07-23"), length.out = 100))
time2 <- sample(seq(as.Date("2020-05-01"), as.Date("2020-09-23"), length.out = 140))
time <- c(time1, time2)
x <- c(rnorm(100), rnorm(140, 1, 1.5))
group <- c(rep("Alpha", 100), rep("Beta", 140))
draw_ts(x, time, 7, group)
```

draw_varimp	<i>Interactive Variable Importance Plot</i>
-------------	---

Description

Plot variable importance using plotly

Usage

```
draw_varimp(
  x,
  names = NULL,
  main = NULL,
  type = c("bar", "line"),
  xlab = NULL,
  ylab = NULL,
  plot_top = 1,
  orientation = "v",
  line_width = 12,
  labelify = TRUE,
  alpha = 1,
  palette = get_palette(getOption("rtemis_palette")),
  mar = NULL,
  font_size = 16,
  axis_font_size = 14,
  theme = choose_theme(getOption("rtemis_theme")),
  showlegend = TRUE,
  filename = NULL,
  file_width = 500,
  file_height = 500,
  file_scale = 1
)
```

Arguments

x	Numeric vector (or coercible to numeric): Input.
names	Vector, string: Names of features.
main	Character: Main title.
type	Character: "bar" or "line".
xlab	Character: x-axis label.
ylab	Character: y-axis label.
plot_top	Numeric: Number of top features to plot if > 1, or the proportion of features to plot if <= 1 (e.g. 1 plots all features).
orientation	Character: "h" or "v".
line_width	Numeric: Line width.

labelify	Logical: If TRUE, labelify feature names.
alpha	Numeric: Transparency.
palette	Character vector: Colors to use.
mar	Vector, numeric, length 4: Plot margins in pixels (NOT inches).
font_size	Integer: Overall font size to use (essentially for the title at this point).
axis_font_size	Integer: Font size to use for axis labels and tick labels.
theme	Theme object.
showlegend	Logical: If TRUE, show legend.
filename	Character: Path to save the plot image.
file_width	Numeric: Width of the saved plot image.
file_height	Numeric: Height of the saved plot image.
file_scale	Numeric: Scale of the saved plot image.

Details

A simple plotly wrapper to plot horizontal barplots, sorted by value, which can be used to visualize variable importance, model coefficients, etc.

Value

plotly object.

Author(s)

EDG

Examples

```
# synthetic data
x <- rnorm(10)
names(x) <- paste0("Feature_", seq(x))
draw_varimp(x)
draw_varimp(x, orientation = "h")
```

draw_volcano	<i>Volcano Plot</i>
--------------	---------------------

Description

Volcano Plot

Usage

```

draw_volcano(
  x,
  pvals,
  xnames = NULL,
  group = NULL,
  x_thresh = 0,
  p_thresh = 0.05,
  p_adjust_method = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "none"),
  p_transform = function(x) -log10(x),
  legend = NULL,
  legend_lo = NULL,
  legend_hi = NULL,
  label_lo = "Low",
  label_hi = "High",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  margin = list(b = 65, l = 65, t = 50, r = 10, pad = 0),
  xlim = NULL,
  ylim = NULL,
  alpha = NULL,
  hline = NULL,
  hline_col = NULL,
  hline_width = 1,
  hline_dash = "solid",
  hline_annotate = NULL,
  hline_annotation_x = 1,
  theme = choose_theme(getOption("rtemis_theme")),
  annotate = TRUE,
  annotate_col = theme[["labs_col"]],
  font_size = 16,
  palette = NULL,
  legend_x_lo = NULL,
  legend_x_hi = NULL,
  legend_y = 0.97,
  annotate_n = 7L,
  ax_lo = NULL,
  ay_lo = NULL,
  ax_hi = NULL,
  ay_hi = NULL,
  annotate_alpha = 0.7,
  hovertext = NULL,
  displayModeBar = "hover",
  filename = NULL,
  file_width = 500,
  file_height = 500,

```

```

    file_scale = 1,
    verbosity = 1L,
    ...
)

```

Arguments

<code>x</code>	Numeric vector: Input values, e.g. log2 fold change, coefficients, etc.
<code>pvals</code>	Numeric vector: p-values.
<code>xnames</code>	Character vector: x names.
<code>group</code>	Optional factor: Used to color code points. If NULL, significant points below <code>x_thresh</code> , non-significant points, and significant points above <code>x_thresh</code> will be plotted with the first, second and third color of <code>palette</code> .
<code>x_thresh</code>	Numeric x-axis threshold separating low from high.
<code>p_thresh</code>	Numeric: p-value threshold of significance.
<code>p_adjust_method</code>	Character: p-value adjustment method. "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". Use "none" for raw p-values.
<code>p_transform</code>	function.
<code>legend</code>	Logical: If TRUE, show legend. If NULL, set to FALSE when <code>group</code> = NULL, otherwise TRUE.
<code>legend_lo</code>	Character: Legend to annotate significant points below the <code>x_thresh</code> .
<code>legend_hi</code>	Character: Legend to annotate significant points above the <code>x_thresh</code> .
<code>label_lo</code>	Character: label for low values.
<code>label_hi</code>	Character: label for high values.
<code>main</code>	Character: Main title.
<code>xlab</code>	Character: x-axis label.
<code>ylab</code>	Character: y-axis label.
<code>margin</code>	Named list of plot margins.
<code>xlim</code>	Numeric vector, length 2: x-axis limits.
<code>ylim</code>	Numeric vector, length 2: y-axis limits.
<code>alpha</code>	Numeric: point transparency.
<code>hline</code>	Numeric: If defined, draw a horizontal line at this y value.
<code>hline_col</code>	Color for hline.
<code>hline_width</code>	Numeric: Width for hline.
<code>hline_dash</code>	Character: Type of line to draw: "solid", "dot", "dash", "longdash", "dashdot", or "longdashdot".
<code>hline_annotate</code>	Character: Text of horizontal line annotation if <code>hline</code> is set.
<code>hline_annotation_x</code>	Numeric: x position to place annotation with paper as reference. 0: to the left of the plot area; 1: to the right of the plot area.

theme	Theme object.
annotate	Logical: If TRUE, annotate significant points.
annotate_col	Color for annotations.
font_size	Integer: Font size.
palette	Character vector: Colors to use. If group is NULL, the first, second and third colors will be used for significant points with negative coefficients, non-significant points, and significant points with positive coefficients, respectively. If group is not NULL, colors will be assigned to groups, in order of appearance.
legend_x_lo	Numeric: x position of legend_lo.
legend_x_hi	Numeric: x position of legend_hi.
legend_y	Numeric: y position for legend_lo and legend_hi.
annotate_n	Integer: Number of significant points to annotate.
ax_lo	Numeric: Sets the x component of the arrow tail about the arrow head for significant points below x_thresh.
ay_lo	Numeric: Sets the y component of the arrow tail about the arrow head for significant points below x_thresh.
ax_hi	Numeric: Sets the x component of the arrow tail about the arrow head for significant points above x_thresh.
ay_hi	Numeric: Sets the y component of the arrow tail about the arrow head for significant points above x_thresh.
annotate_alpha	Numeric: Transparency for annotations.
hovertext	Character vector: Text to display on hover.
displayModeBar	Logical: If TRUE, display plotly mode bar.
filename	Character: Path to save the plot image.
file_width	Numeric: Width of the saved plot image.
file_height	Numeric: Height of the saved plot image.
file_scale	Numeric: Scale of the saved plot image.
verbosity	Integer: Verbosity level.
...	Additional arguments passed to draw_scatter .

Value

plotly object.

Author(s)

EDG

Examples

```
set.seed(2019)
y <- rnormmat(500, 500, return_df = TRUE)
x <- data.frame(x = y[, 3] + y[, 5] - y[, 9] + y[, 15] + rnorm(500))
mod <- massGLM(x, y)
draw_volcano(summary(mod)[["Coefficient_x"]], summary(mod)[["p_value_x"]])
```

draw_xt

*Plot timeseries data***Description**

Plot timeseries data

Usage

```
draw_xt(
  x,
  y,
  x2 = NULL,
  y2 = NULL,
  which_xy = NULL,
  which_xy2 = NULL,
  shade_bin = NULL,
  shade_interval = NULL,
  shade_col = NULL,
  shade_x = NULL,
  shade_name = "",
  shade_showlegend = FALSE,
  ynames = NULL,
  y2names = NULL,
  xlab = NULL,
  ylab = NULL,
  y2lab = NULL,
  xunits = NULL,
  yunits = NULL,
  y2units = NULL,
  yunits_col = NULL,
  y2units_col = NULL,
  zt = NULL,
  show_zt = TRUE,
  show_zt_every = NULL,
  zt_nticks = 18L,
  main = NULL,
  main_y = 1,
  main_yanchor = "bottom",
  x_nticks = 0,
  y_nticks = 0,
  show_rangeslider = NULL,
  slider_start = NULL,
  slider_end = NULL,
  theme = choose_theme(getOption("rtemis_theme")),
  palette = get_palette(getOption("rtemis_palette")),
  font_size = 16,
```

```

yfill = "none",
y2fill = "none",
fill_alpha = 0.2,
yline_width = 2,
y2line_width = 2,
x_showspikes = TRUE,
spike_dash = "solid",
spike_col = NULL,
x_spike_thickness = -2,
tickfont_size = 16,
x_tickmode = "auto",
x_tickvals = NULL,
x_ticktext = NULL,
x_tickangle = NULL,
legend_x = 0,
legend_y = 1.1,
legend_xanchor = "left",
legend_yanchor = "top",
legend_orientation = "h",
margin = list(l = 75, r = 75, b = 75, t = 75),
x_standoff = 20L,
y_standoff = 20L,
y2_standoff = 20L,
hovermode = "x",
displayModeBar = TRUE,
modeBar_file_format = "svg",
scrollZoom = TRUE,
filename = NULL,
file_width = 960,
file_height = 500,
file_scale = 1
)

```

Arguments

<code>x</code>	Datetime vector or list of vectors.
<code>y</code>	Numeric vector or named list of vectors: y-axis data.
<code>x2</code>	Datetime vector or list of vectors, optional: must be provided if y2 does not correspond to values in x. A single x-axis will be drawn for all values in x and x2.
<code>y2</code>	Numeric vector, optional: If provided, a second y-axis will be added to the right side of the plot.
<code>which_xy</code>	Integer vector: Indices of x and y to plot. If not provided, will select up to the first two x-y traces.
<code>which_xy2</code>	Integer vector: Indices of x2 and y2 to plot. If not provided, will select up to the first two x2-y2 traces.

shade_bin	Integer vector {0, 1}: Time points in x to shade on the plot. For example, if there are 10 time points in x, and you want to shade time points 3 to 7, shade_bin = c(0, 0, 1, 1, 1, 1, 1, 0, 0, 0). Only set shade_bin or shade_interval, not both.
shade_interval	List of numeric vectors: Intervals to shade on the plot. Only set shade_bin or shade_interval, not both.
shade_col	Color: Color to shade intervals.
shade_x	Numeric vector: x-values to use for shading.
shade_name	Character: Name for shaded intervals.
shade_showlegend	Logical: If TRUE, show legend for shaded intervals.
ynames	Character vector, optional: Names for each vector in y.
y2names	Character vector, optional: Names for each vector in y2.
xlab	Character: x-axis label.
ylab	Character: y-axis label.
y2lab	Character: y2-axis label.
xunits	Character: x-axis units.
yunits	Character: y-axis units.
y2units	Character: y2-axis units.
yunits_col	Color for y-axis units.
y2units_col	Color for y2-axis units.
zt	Numeric vector: Zeitgeber time. If provided, will be shown on the x-axis instead of x. To be used only with a single x vector and no x2.
show_zt	Logical: If TRUE, show zt on x-axis, if zt is provided.
show_zt_every	Optional integer: Show zt every show_zt_every ticks. If NULL, will be calculated to be x_nticks +/- 1 if x_nticks is not 0, otherwise 12 +/- 1.
zt_nticks	Integer: Number of zt ticks to show. Only used if show_zt_every is NULL. The actual number of ticks shown will depend on the periodicity of zt, so that zt = 0 is always included.
main	Character: Main title.
main_y	Numeric: Y position of main title.
main_ylanchor	Character: "top", "middle", "bottom".
x_nticks	Integer: Number of ticks on x-axis.
y_nticks	Integer: Number of ticks on y-axis.
show_rangeslider	Logical: If TRUE, show a range slider.
slider_start	Numeric: Start of range slider.
slider_end	Numeric: End of range slider.
theme	Theme object.
palette	Character vector: Colors to be used to draw each vector in y and y2, in order.

font_size	Numeric: Font size for text.
yfill	Character: Fill type for y-axis: "none", "tozero", "tonexty".
y2fill	Character: Fill type for y2-axis: "none", "tozero", "tonexty".
fill_alpha	Numeric: Fill opacity for y-axis.
yline_width	Numeric: Line width for y-axis lines.
y2line_width	Numeric: Line width for y2-axis lines.
x_showspikes	Logical: If TRUE, show spikes on x-axis.
spike_dash	Character: Dash type for spikes: "solid", "dot", "dash", "longdash", "dashdot", "longdashdot".
spike_col	Color for spikes.
x_spike_thickness	Numeric: Thickness of spikes. -2 avoids drawing border around spikes.
tickfont_size	Numeric: Font size for tick labels.
x_tickmode	Character: "auto", "linear", "array".
x_tickvals	Numeric vector: Tick positions.
x_ticktext	Character vector: Tick labels.
x_tickangle	Numeric: Angle of tick labels.
legend_x	Numeric: X position of legend.
legend_y	Numeric: Y position of legend.
legend_xanchor	Character: "left", "center", "right".
legend_yanchor	Character: "top", "middle", "bottom".
legend_orientation	Character: "v" for vertical, "h" for horizontal.
margin	Named list with 4 numeric values: "l", "r", "t", "b" for left, right, top, bottom margins.
x_standoff	Numeric: Distance from x-axis to x-axis label.
y_standoff	Numeric: Distance from y-axis to y-axis label.
y2_standoff	Numeric: Distance from y2-axis to y2-axis label.
hovermode	Character: "closest", "x", "x unified".
displayModeBar	Logical: If TRUE, display plotly mode bar.
modeBar_file_format	Character: "png", "svg", "jpeg", "webp", "pdf": file format for mode bar image export.
scrollZoom	Logical: If TRUE, enable zooming by scrolling.
filename	Character: Path to save the plot image.
file_width	Numeric: Width of the saved plot image.
file_height	Numeric: Height of the saved plot image.
file_scale	Numeric: Scale of the saved plot image.

Value

plotly object.

Author(s)

EDG

Examples

```
datetime <- seq(
  as.POSIXct("2020-01-01 00:00"),
  as.POSIXct("2020-01-02 00:00"),
  by = "hour"
)
df <- data.frame(
  datetime = datetime,
  value1 = rnorm(length(datetime)),
  value2 = rnorm(length(datetime))
)
draw_xt(x = df[, 1], y = df[, 2:3])
```

get_palette

Get Color Palette

Description

get_palette() returns a color palette (character vector of colors). Without arguments, prints names of available color palettes. Each palette is a named list of hexadecimal color definitions which can be used with any graphics function.

Usage

```
get_palette(palette = NULL, verbosity = 1L)
```

Arguments

palette	Character: Name of palette to return. If NULL, available palette names are printed and no palette is returned.
verbosity	Integer: Verbosity level.

Value

Character vector of colors for the specified palette, or invisibly returns list of available palettes if palette = NULL.

Author(s)

EDG

Examples

```
# Print available palettes
get_palette()
# Get the Imperial palette
get_palette("imperial")
```

```
previewcolor
```

```
Preview color
```

Description

Preview one or multiple colors using little rhombi with their little labels up top

Usage

```
previewcolor(
  x,
  main = NULL,
  bg = "#333333",
  main_col = "#b3b3b3",
  main_x = 0.7,
  main_y = 0.2,
  main_adj = 0,
  main_cex = 0.9,
  main_font = 2,
  width = NULL,
  xlim = NULL,
  ylim = c(0, 2.2),
  asp = 1,
  labels_y = 1.55,
  label_cex = NULL,
  mar = c(0, 0, 0, 1),
  filename = NULL,
  pdf_width = 8,
  pdf_height = 2.5
)
```

Arguments

<code>x</code>	Color, vector: One or more colors that R understands
<code>main</code>	Character: Title. If NULL, set to <code>deparse(substitute(x))</code> .
<code>bg</code>	Background color.
<code>main_col</code>	Color: Title color
<code>main_x</code>	Float: x coordinate for main.
<code>main_y</code>	Float: y coordinate for main.

main_adj	Float: adj argument to mtext for main.
main_cex	Float: character expansion factor for main.
main_font	Integer, 1 or 2: Weight of main 1: regular, 2: bold.
width	Float: Plot width. If NULL, set automatically.
xlim	Vector, length 2: x-axis limits. If NULL, set automatically.
ylim	Vector, length 2: y-axis limits.
asp	Float: Plot aspect ratio.
labels_y	Float: y coord for labels (rhombi are fixed and range y .5 - 1.5).
label_cex	Float: Character expansion for labels. If NULL, calculated automatically based on length of x.
mar	Numeric vector, length 4: margin size.
filename	Character: Path to save plot as PDF.
pdf_width	Numeric: Width of PDF in inches.
pdf_height	Numeric: Height of PDF in inches.

Value

Nothing, prints plot.

Author(s)

EDG

Examples

```
previewcolor(get_palette("rtms"))
```

theme_black	<i>Themes for draw_* functions</i>
-------------	------------------------------------

Description

Themes for draw_* functions

Usage

```
theme_black(
  bg = "#000000",
  plot_bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
```

```

box_col = fg,
box_alpha = 1,
box_lty = 1,
box_lwd = 0.5,
grid = FALSE,
grid_nx = NULL,
grid_ny = NULL,
grid_col = fg,
grid_alpha = 0.2,
grid_lty = 1,
grid_lwd = 1,
axes_visible = TRUE,
axes_col = "transparent",
tick_col = fg,
tick_alpha = 0.5,
tick_labels_col = fg,
tck = -0.01,
tcl = NA,
x_axis_side = 1,
y_axis_side = 2,
labs_col = fg,
x_axis_line = 0,
x_axis_las = 0,
x_axis_padj = -1.1,
x_axis_hadj = 0.5,
y_axis_line = 0,
y_axis_las = 1,
y_axis_padj = 0.5,
y_axis_hadj = 0.5,
xlab_line = 1.4,
ylab_line = 2,
zerolines = TRUE,
zerolines_col = fg,
zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)

theme_blackgrid(
  bg = "#000000",
  plot_bg = "transparent",
  fg = "#ffffff",
  pch = 16,

```

```

    cex = 1,
    lwd = 2,
    bty = "n",
    box_col = fg,
    box_alpha = 1,
    box_lty = 1,
    box_lwd = 0.5,
    grid = TRUE,
    grid_nx = NULL,
    grid_ny = NULL,
    grid_col = fg,
    grid_alpha = 0.2,
    grid_lty = 1,
    grid_lwd = 1,
    axes_visible = TRUE,
    axes_col = "transparent",
    tick_col = fg,
    tick_alpha = 1,
    tick_labels_col = fg,
    tck = -0.01,
    tcl = NA,
    x_axis_side = 1,
    y_axis_side = 2,
    labs_col = fg,
    x_axis_line = 0,
    x_axis_las = 0,
    x_axis_padj = -1.1,
    x_axis_hadj = 0.5,
    y_axis_line = 0,
    y_axis_las = 1,
    y_axis_padj = 0.5,
    y_axis_hadj = 0.5,
    xlab_line = 1.4,
    ylab_line = 2,
    zerolines = TRUE,
    zerolines_col = fg,
    zerolines_alpha = 0.5,
    zerolines_lty = 1,
    zerolines_lwd = 1,
    main_line = 0.25,
    main_adj = 0,
    main_font = 2,
    main_col = fg,
    font_family = getOption("rtemis_font", "Helvetica")
)

theme_blackigrid(
  bg = "#000000",

```

```
plot_bg = "#1A1A1A",
fg = "#ffffff",
pch = 16,
cex = 1,
lwd = 2,
bty = "n",
box_col = fg,
box_alpha = 1,
box_lty = 1,
box_lwd = 0.5,
grid = TRUE,
grid_nx = NULL,
grid_ny = NULL,
grid_col = bg,
grid_alpha = 1,
grid_lty = 1,
grid_lwd = 1,
axes_visible = TRUE,
axes_col = "transparent",
tick_col = fg,
tick_alpha = 1,
tick_labels_col = fg,
tck = -0.01,
tcl = NA,
x_axis_side = 1,
y_axis_side = 2,
labs_col = fg,
x_axis_line = 0,
x_axis_las = 0,
x_axis_padj = -1.1,
x_axis_hadj = 0.5,
y_axis_line = 0,
y_axis_las = 1,
y_axis_padj = 0.5,
y_axis_hadj = 0.5,
xlab_line = 1.4,
ylab_line = 2,
zerolines = TRUE,
zerolines_col = fg,
zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)
```

```
theme_darkgray(  
  bg = "#121212",  
  plot_bg = "transparent",  
  fg = "ffffff",  
  pch = 16,  
  cex = 1,  
  lwd = 2,  
  bty = "n",  
  box_col = fg,  
  box_alpha = 1,  
  box_lty = 1,  
  box_lwd = 0.5,  
  grid = FALSE,  
  grid_nx = NULL,  
  grid_ny = NULL,  
  grid_col = fg,  
  grid_alpha = 0.2,  
  grid_lty = 1,  
  grid_lwd = 1,  
  axes_visible = TRUE,  
  axes_col = "transparent",  
  tick_col = fg,  
  tick_alpha = 0.5,  
  tick_labels_col = fg,  
  tck = -0.01,  
  tcl = NA,  
  x_axis_side = 1,  
  y_axis_side = 2,  
  labs_col = fg,  
  x_axis_line = 0,  
  x_axis_las = 0,  
  x_axis_padj = -1.1,  
  x_axis_hadj = 0.5,  
  y_axis_line = 0,  
  y_axis_las = 1,  
  y_axis_padj = 0.5,  
  y_axis_hadj = 0.5,  
  xlab_line = 1.4,  
  ylab_line = 2,  
  zerolines = TRUE,  
  zerolines_col = fg,  
  zerolines_alpha = 0.5,  
  zerolines_lty = 1,  
  zerolines_lwd = 1,  
  main_line = 0.25,  
  main_adj = 0,  
  main_font = 2,
```

```

    main_col = fg,
    font_family = getOption("rtemis_font", "Helvetica")
)

```

```

theme_darkgraygrid(
  bg = "#121212",
  plot_bg = "transparent",
  fg = "#ffffff",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box_col = fg,
  box_alpha = 1,
  box_lty = 1,
  box_lwd = 0.5,
  grid = TRUE,
  grid_nx = NULL,
  grid_ny = NULL,
  grid_col = "#404040",
  grid_alpha = 1,
  grid_lty = 1,
  grid_lwd = 1,
  axes_visible = TRUE,
  axes_col = "transparent",
  tick_col = "#00000000",
  tick_alpha = 1,
  tick_labels_col = fg,
  tck = -0.01,
  tcl = NA,
  x_axis_side = 1,
  y_axis_side = 2,
  labs_col = fg,
  x_axis_line = 0,
  x_axis_las = 0,
  x_axis_padj = -1.1,
  x_axis_hadj = 0.5,
  y_axis_line = 0,
  y_axis_las = 1,
  y_axis_padj = 0.5,
  y_axis_hadj = 0.5,
  xlab_line = 1.4,
  ylab_line = 2,
  zerolines = TRUE,
  zerolines_col = fg,
  zerolines_alpha = 0.5,
  zerolines_lty = 1,
  zerolines_lwd = 1,

```

```
    main_line = 0.25,  
    main_adj = 0,  
    main_font = 2,  
    main_col = fg,  
    font_family = getOption("rtemis_font", "Helvetica")  
  )
```

```
theme_darkgrayigrid(  
  bg = "#121212",  
  plot_bg = "#202020",  
  fg = "#ffffff",  
  pch = 16,  
  cex = 1,  
  lwd = 2,  
  bty = "n",  
  box_col = fg,  
  box_alpha = 1,  
  box_lty = 1,  
  box_lwd = 0.5,  
  grid = TRUE,  
  grid_nx = NULL,  
  grid_ny = NULL,  
  grid_col = bg,  
  grid_alpha = 1,  
  grid_lty = 1,  
  grid_lwd = 1,  
  axes_visible = TRUE,  
  axes_col = "transparent",  
  tick_col = "transparent",  
  tick_alpha = 1,  
  tick_labels_col = fg,  
  tck = -0.01,  
  tcl = NA,  
  x_axis_side = 1,  
  y_axis_side = 2,  
  labs_col = fg,  
  x_axis_line = 0,  
  x_axis_las = 0,  
  x_axis_padj = -1.1,  
  x_axis_hadj = 0.5,  
  y_axis_line = 0,  
  y_axis_las = 1,  
  y_axis_padj = 0.5,  
  y_axis_hadj = 0.5,  
  xlab_line = 1.4,  
  ylab_line = 2,  
  zerolines = TRUE,  
  zerolines_col = fg,  
)
```

```

zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)

```

```

theme_white(
  bg = "#ffffff",
  plot_bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box_col = fg,
  box_alpha = 1,
  box_lty = 1,
  box_lwd = 0.5,
  grid = FALSE,
  grid_nx = NULL,
  grid_ny = NULL,
  grid_col = fg,
  grid_alpha = 1,
  grid_lty = 1,
  grid_lwd = 1,
  axes_visible = TRUE,
  axes_col = "transparent",
  tick_col = fg,
  tick_alpha = 0.5,
  tick_labels_col = fg,
  tck = -0.01,
  tcl = NA,
  x_axis_side = 1,
  y_axis_side = 2,
  labs_col = fg,
  x_axis_line = 0,
  x_axis_las = 0,
  x_axis_padj = -1.1,
  x_axis_hadj = 0.5,
  y_axis_line = 0,
  y_axis_las = 1,
  y_axis_padj = 0.5,
  y_axis_hadj = 0.5,
  xlab_line = 1.4,

```

```
ylab_line = 2,  
zerolines = TRUE,  
zerolines_col = fg,  
zerolines_alpha = 0.5,  
zerolines_lty = 1,  
zerolines_lwd = 1,  
main_line = 0.25,  
main_adj = 0,  
main_font = 2,  
main_col = fg,  
font_family = getOption("rtemis_font", "Helvetica")  
)
```

```
theme_whitegrid(  
  bg = "#ffffff",  
  plot_bg = "transparent",  
  fg = "#000000",  
  pch = 16,  
  cex = 1,  
  lwd = 2,  
  bty = "n",  
  box_col = fg,  
  box_alpha = 1,  
  box_lty = 1,  
  box_lwd = 0.5,  
  grid = TRUE,  
  grid_nx = NULL,  
  grid_ny = NULL,  
  grid_col = "#c0c0c0",  
  grid_alpha = 1,  
  grid_lty = 1,  
  grid_lwd = 1,  
  axes_visible = TRUE,  
  axes_col = "transparent",  
  tick_col = "#00000000",  
  tick_alpha = 1,  
  tick_labels_col = fg,  
  tck = -0.01,  
  tcl = NA,  
  x_axis_side = 1,  
  y_axis_side = 2,  
  labs_col = fg,  
  x_axis_line = 0,  
  x_axis_las = 0,  
  x_axis_padj = -1.1,  
  x_axis_hadj = 0.5,  
  y_axis_line = 0,  
  y_axis_las = 1,
```

```

y_axis_padj = 0.5,
y_axis_hadj = 0.5,
xlab_line = 1.4,
ylab_line = 2,
zerolines = TRUE,
zerolines_col = fg,
zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)

```

```

theme_whitegrid(
  bg = "#ffffff",
  plot_bg = "#E6E6E6",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box_col = fg,
  box_alpha = 1,
  box_lty = 1,
  box_lwd = 0.5,
  grid = TRUE,
  grid_nx = NULL,
  grid_ny = NULL,
  grid_col = bg,
  grid_alpha = 1,
  grid_lty = 1,
  grid_lwd = 1,
  axes_visible = TRUE,
  axes_col = "transparent",
  tick_col = "transparent",
  tick_alpha = 1,
  tick_labels_col = fg,
  tck = -0.01,
  tcl = NA,
  x_axis_side = 1,
  y_axis_side = 2,
  labs_col = fg,
  x_axis_line = 0,
  x_axis_las = 0,
  x_axis_padj = -1.1,

```

```

    x_axis_hadj = 0.5,
    y_axis_line = 0,
    y_axis_las = 1,
    y_axis_padj = 0.5,
    y_axis_hadj = 0.5,
    xlab_line = 1.4,
    ylab_line = 2,
    zerolines = TRUE,
    zerolines_col = fg,
    zerolines_alpha = 0.5,
    zerolines_lty = 1,
    zerolines_lwd = 1,
    main_line = 0.25,
    main_adj = 0,
    main_font = 2,
    main_col = fg,
    font_family = getOption("rtemis_font", "Helvetica")
)

```

```

theme_lightgraygrid(
  bg = "#dfdfdf",
  plot_bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box_col = fg,
  box_alpha = 1,
  box_lty = 1,
  box_lwd = 0.5,
  grid = TRUE,
  grid_nx = NULL,
  grid_ny = NULL,
  grid_col = "#c0c0c0",
  grid_alpha = 1,
  grid_lty = 1,
  grid_lwd = 1,
  axes_visible = TRUE,
  axes_col = "transparent",
  tick_col = "#00000000",
  tick_alpha = 1,
  tick_labels_col = fg,
  tck = -0.01,
  tcl = NA,
  x_axis_side = 1,
  y_axis_side = 2,
  labs_col = fg,

```

```

x_axis_line = 0,
x_axis_las = 0,
x_axis_padj = -1.1,
x_axis_hadj = 0.5,
y_axis_line = 0,
y_axis_las = 1,
y_axis_padj = 0.5,
y_axis_hadj = 0.5,
xlab_line = 1.4,
ylab_line = 2,
zerolines = TRUE,
zerolines_col = fg,
zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)

```

```

theme_mediumgraygrid(
  bg = "#b3b3b3",
  plot_bg = "transparent",
  fg = "#000000",
  pch = 16,
  cex = 1,
  lwd = 2,
  bty = "n",
  box_col = fg,
  box_alpha = 1,
  box_lty = 1,
  box_lwd = 0.5,
  grid = TRUE,
  grid_nx = NULL,
  grid_ny = NULL,
  grid_col = "#d0d0d0",
  grid_alpha = 1,
  grid_lty = 1,
  grid_lwd = 1,
  axes_visible = TRUE,
  axes_col = "transparent",
  tick_col = "#00000000",
  tick_alpha = 1,
  tick_labels_col = fg,
  tck = -0.01,
  tcl = NA,

```

```

x_axis_side = 1,
y_axis_side = 2,
labs_col = fg,
x_axis_line = 0,
x_axis_las = 0,
x_axis_padj = -1.1,
x_axis_hadj = 0.5,
y_axis_line = 0,
y_axis_las = 1,
y_axis_padj = 0.5,
y_axis_hadj = 0.5,
xlab_line = 1.4,
ylab_line = 2,
zerolines = TRUE,
zerolines_col = fg,
zerolines_alpha = 0.5,
zerolines_lty = 1,
zerolines_lwd = 1,
main_line = 0.25,
main_adj = 0,
main_font = 2,
main_col = fg,
font_family = getOption("rtemis_font", "Helvetica")
)

```

Arguments

bg	Color: Figure background.
plot_bg	Color: Plot region background.
fg	Color: Foreground color used as default for multiple elements like axes and labels, which can be defined separately.
pch	Integer: Point character.
cex	Float: Character expansion factor.
lwd	Float: Line width.
bty	Character: Box type: "o", "l", "7", "c", "u", or "]", or "n".
box_col	Box color if bty != "n".
box_alpha	Float: Box alpha.
box_lty	Integer: Box line type.
box_lwd	Float: Box line width.
grid	Logical: If TRUE, draw grid in plot regions.
grid_nx	Integer: N of vertical grid lines.
grid_ny	Integer: N of horizontal grid lines.
grid_col	Grid color.
grid_alpha	Float: Grid alpha.

grid_lty	Integer: Grid line type.
grid_lwd	Float: Grid line width.
axes_visible	Logical: If TRUE, draw axes.
axes_col	Axes colors.
tick_col	Tick color.
tick_alpha	Float: Tick alpha.
tick_labels_col	Tick labels' color.
tck	graphics::parr's tck argument: Tick length, can be negative.
tcl	graphics::parr's tcl argument.
x_axis_side	Integer: Side to place x-axis.
y_axis_side	Integer: Side to place y-axis.
labs_col	Labels' color.
x_axis_line	Numeric: graphics::axis's line argument for the x-axis.
x_axis_las	Numeric: graphics::axis's las argument for the x-axis.
x_axis_padj	Numeric: x-axis' padj: Adjustment for the x-axis tick labels' position.
x_axis_hadj	Numeric: x-axis' hadj.
y_axis_line	Numeric: graphics::axis's line argument for the y-axis.
y_axis_las	Numeric: graphics::axis's las argument for the y-axis.
y_axis_padj	Numeric: y-axis' padj.
y_axis_hadj	Numeric: y-axis' hadj.
xlab_line	Numeric: Line to place xlab.
ylab_line	Numeric: Line to place ylab.
zerolines	Logical: If TRUE, draw lines on x = 0, y = 0, if within plot limits.
zerolines_col	Zerolines color.
zerolines_alpha	Float: Zerolines alpha.
zerolines_lty	Integer: Zerolines line type.
zerolines_lwd	Float: Zerolines line width.
main_line	Float: How many lines away from the plot region to draw title.
main_adj	Float: How to align title.
main_font	Integer: 1: Regular, 2: Bold.
main_col	Title color.
font_family	Character: Font to be used throughout plot.

Value

Theme object.

Examples

```
theme <- theme_black(font_family = "Geist")
theme
```

uniprot_get	<i>Get protein sequence from UniProt</i>
-------------	--

Description

Get protein sequence from UniProt

Usage

```
uniprot_get(  
  accession,  
  baseURL = "https://rest.uniprot.org/uniprotkb",  
  verbosity = 1  
)
```

Arguments

accession	Character: UniProt Accession number - e.g. "Q9UMX9"
baseURL	Character: UniProt rest API base URL.
verbosity	Integer: Verbosity level.

Value

List with three elements: Identifier, Annotation, and Sequence.

Author(s)

E.D. Gennatas

Examples

```
## Not run:  
# This gets the sequence from uniprot.org by default  
mapt <- uniprot_get("Q9UMX9")  
  
## End(Not run)
```

xt_example	<i>Example longitudinal dataset</i>
------------	-------------------------------------

Description

A small synthetic dataset demonstrating various participation patterns in longitudinal data, suitable for examples with [xtdescribe](#).

Usage

xt_example

Format

A data frame with 30 rows and 4 variables:

- patient_id** Integer: Patient identifier (1-10).
- year** Integer: Year of measurement (2020-2024).
- blood_pressure** Numeric: Systolic blood pressure measurement.
- treatment** Character: Treatment group ("A" or "B").

Details

This dataset includes 10 patients measured at up to 5 time points (years 2020-2024). The dataset demonstrates various participation patterns typical in longitudinal studies:

- Complete participation (all time points)
- Early dropout
- Late entry
- Intermittent participation
- Single time point participation

Examples

```
data(xt_example)
head(xt_example)
summary(xt_example)
```

xtdescribe	<i>Describe longitudinal dataset</i>
------------	--------------------------------------

Description

This function emulates the xtdescribe function in Stata.

Usage

```
xtdescribe(x, id_col = 1, time_col = 2, n_patterns = 9)
```

Arguments

x	data.frame: Longitudinal data with ID and time variables.
id_col	Integer: The column position of the ID variable.
time_col	Integer: The column position of the time variable.
n_patterns	Integer: The number of patterns to display.

Value

data.frame: Summary of participation patterns, returned invisibly.

Author(s)

EDG

Examples

```
# Load example longitudinal dataset
data(xt_example)

# Describe the longitudinal structure
xtdescribe(xt_example)
```

Index

- * **datasets**
 - xt_example, [84](#)
- available_draw, [3](#)
- available_themes, [3](#)
- choose_theme, [4](#)
- col2grayscale, [5](#)
- color_adjust, [5](#)
- ddSci, [54](#)
- draw_3Dscatter, [6](#)
- draw_bar, [9](#), [41](#)
- draw_box, [12](#)
- draw_calibration, [17](#)
- draw_confusion, [19](#)
- draw_dist, [20](#)
- draw_fit, [24](#)
- draw_graphD3, [25](#)
- draw_graphjs, [26](#)
- draw_heatmap, [28](#)
- draw_leaflet, [31](#)
- draw_pie, [33](#)
- draw_protein, [35](#)
- draw_pvals, [41](#)
- draw_roc, [42](#)
- draw_scatter, [18](#), [24](#), [43](#), [44](#), [52](#), [53](#), [57](#), [62](#)
- draw_spectrogram, [49](#)
- draw_survfit, [52](#)
- draw_table, [54](#)
- draw_ts, [55](#)
- draw_varimp, [58](#)
- draw_volcano, [59](#)
- draw_xt, [63](#)
- get_palette, [67](#)
- labelify, [15](#)
- p.adjust, [41](#)
- previewcolor, [68](#)
- rtemis::classification_metrics(), [19](#)
- survival::survfit, [53](#)
- theme_black, [69](#)
- theme_blackgrid(theme_black), [69](#)
- theme_blackigrid(theme_black), [69](#)
- theme_darkgray(theme_black), [69](#)
- theme_darkgraygrid(theme_black), [69](#)
- theme_darkgrayigrid(theme_black), [69](#)
- theme_lightgraygrid(theme_black), [69](#)
- theme_mediumgraygrid(theme_black), [69](#)
- theme_white(theme_black), [69](#)
- theme_whitegrid(theme_black), [69](#)
- theme_whiteigrid(theme_black), [69](#)
- uniprot_get, [83](#)
- xt_example, [84](#)
- xtdescribe, [84](#), [85](#)