Package 'ggforge'

December 5, 2025

Title Comprehensive Visualization Toolkit for Biomedical Research

Version 1.0.0

Description A comprehensive and elegant visualization framework designed for biomedical and bioinformatics research. Built on 'ggplot2' with a modern modular architecture, it provides 40+ publication-ready plotting functions covering statistical graphics, enrichment analysis (GO, KEGG, GSEA), single-cell and spatial transcriptomics (dimensionality reduction, velocity fields), multi-omics integration (heatmaps, chord diagrams, networks), survival analysis (Kaplan-Meier curves), genomics (volcano plots, Manhattan plots), and advanced visualizations (sankey diagrams, upset plots, word clouds). Features a unified API design with intelligent type detection, publication-grade themes, extensive color palettes, and flexible customization options. Streamlines the creation of complex multi-panel figures with minimal code while maintaining full control over aesthetics.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports circlize,
cowplot,
dplyr,
forcats,
ggnewscale,
ggplot2,
ggrepel,
ggridges,
glue,
gridtext,
gtable,
methods,
patchwork,
reshape2,
rlang,
scales,
stringr,
tidyr,
zoo

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown,
Matrix,
alluvial,
ComplexHeatmap,
cluster,
clustree,
ggwordcloud,
ggalluvial,
ggVennDiagram (>= 1.5.0),
ggupset,
ggpubr,
ggforce,
ggraph,
ggridges,
ggmanh,
qqplotr,
hexbin,
igraph,
iNEXT,
scattermore,
sf,
terra,
concaveman,
plotROC,
OptimalCutpoints,
proxyC,
metR

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

Contents

`AreaPlot` *Area Plot*

Description

A plot showing how one or more groups' numeric values change over the progression of another variable.

Usage

```
AreaPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	A character string of the column name to plot on the x-axis. A character/factor column is expected.
y	A character string of the column name to plot on the y-axis. A numeric column is expected. If NULL, the count of the x-axis column will be used.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.

group_by	A character vector of column names to fill the area plot by. If NULL, the plot will be filled by the first color of the palette. If multiple columns are provided, the columns will be concatenated with group_by_sep and used as the fill column.
group_by_sep	A character string to separate the columns in group_by.
group_name	A character string to name the legend of fill.
scale_y	A logical value to scale the y-axis by the total number in each x-axis group.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_nrow	Number of rows in facet layout
facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
x_text_angle	Angle for x-axis text
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Whether to keep empty factor levels
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
  x = rep(c("A", "B", "C", "D"), 2),
  y = c(1, 3, 6, 4, 2, 5, 7, 8),
  group = rep(c("F1", "F2"), each = 4),
  split = rep(c("X", "Y"), 4)
)
AreaPlot(data, x = "x", y = "y", group_by = "group")
AreaPlot(data,
  x = "x", y = "y", group_by = "group",
  scale_y = TRUE
)
AreaPlot(data, x = "x", y = "y", split_by = "group")
AreaPlot(data, x = "x", y = "y", split_by = "group", palette = c(F1 = "Blues", F2 = "Reds"))
AreaPlot(data,
  x = "x", y = "y", group_by = "group", split_by = "split",
  legend.direction = c(X = "horizontal", Y = "vertical"),
  legend.position = c(X = "top", Y = "right")
)
```

BarPlot *Bar Plot*

Description

Create a bar plot with optional grouping, splitting, and faceting.

Bar plots are useful for comparing values across categories. This function supports both simple bar plots and grouped bar plots with stacking or dodging.

Usage

```
BarPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  scale_y = FALSE,
  flip = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  fill_by_x_if_no_group = TRUE,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  label = NULL,
  label_nudge = 0.02,
  label_fg = "black",
```

```
  label_size = 4,
  label_bg = "white",
  label_bg_r = 0.1,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  line_name = NULL,
  add_trend = FALSE,
  trend_color = "black",
  trend_linewidth = 1,
  trend_ptsize = 2.5,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  x_text_angle = 0,
  aspect.ratio = 1,
  y_min = NULL,
  y_max = NULL,
  position = "auto",
  position_dodge_preserve = "total",
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  expand = waiver(),
  width = waiver(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column for x-axis (categorical). Will be converted to factor.

x_sep	Separator for concatenating multiple x columns.
y	Column for y-axis (numeric). If NULL, counts will be used.
scale_y	Scale y values to proportions within each x group (only for grouped plots).
flip	Flip x and y axes.
group_by	Column(s) for grouping bars (creates stacked or dodged bars).
group_by_sep	Separator for concatenating multiple group columns.
group_name	Legend title for groups.
fill_by_x_if_no_group	Fill bars by x values when no grouping (default TRUE).
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
label	Column to use for labels, or TRUE to use y values.
label_nudge	Vertical distance for label positioning.
label_fg	Label text color.
label_size	Label text size.
label_bg	Label background color.
label_bg_r	Label background corner radius.
add_bg	Add background stripes for x categories.
bg_palette	Palette for background stripes.
bg_palcolor	Custom colors for background stripes.
bg_alpha	Transparency for background stripes.
add_line	Add horizontal reference line at specified y value.
line_color	Color of reference line.
line_width	Width of reference line.
line_type	Line type (1=solid, 2=dashed, etc.).
line_name	Legend label for reference line.
add_trend	Add trend line connecting bar tops.
trend_color	Color of trend line (NULL uses group colors).
trend_linewidth	Width of trend line.
trend_ptsize	Size of trend line points.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)

x_text_angle	Angle for x-axis text labels
aspect.ratio	Aspect ratio of plot panel
y_min	Minimum y-axis value.
y_max	Maximum y-axis value.
position	Position adjustment: "auto", "stack", "dodge", or "fill".
position_dodge_preserve	Preserve "total" or "single" width when dodging.
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Keep empty factor levels
expand	Plot area expansion (CSS-like: top, right, bottom, left).
width	Bar width.
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, list of plots, or combined plots

Examples

```
# Simple bar plot (counts)
data <- data.frame(
  category = c("A", "B", "C", "D", "A", "B")
)
BarPlot(data, x = "category")

# Bar plot with values
data <- data.frame(
  category = c("A", "B", "C", "D"),
  value = c(10, 15, 8, 12)
)
BarPlot(data, x = "category", y = "value")
```

```
# Grouped bar plot (dodged)
data <- data.frame(
  category = rep(c("A", "B", "C"), 2),
  value = c(10, 15, 8, 12, 18, 14),
  group = rep(c("G1", "G2"), each = 3)
)
BarPlot(data, x = "category", y = "value", group_by = "group")

# Stacked bar plot
BarPlot(data, x = "category", y = "value", group_by = "group", position = "stack")

# With labels and customization
BarPlot(
  data,
  x = "category", y = "value", group_by = "group",
  label = TRUE, palette = "Set2", flip = TRUE
)

# With splits (multiple plots)
data$experiment <- rep(c("Exp1", "Exp2"), 3)
BarPlot(data, x = "category", y = "value", split_by = "experiment")

# With faceting (single plot, multiple panels)
BarPlot(data, x = "category", y = "value", facet_by = "experiment")
```

BoxPlot *Box Plot*

Description

Create box plots with optional grouping, faceting, and statistical comparisons. Box plots display the distribution of continuous data through their quartiles, showing the median, interquartile range, and potential outliers.

Usage

```
BoxPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  keep_empty = FALSE,
  group_by = NULL,
  group_by_sep = "_",
```

```
    group_name = NULL,
    paired_by = NULL,
    x_text_angle = NULL,
    step_increase = 0.1,
    fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
    fill_reverse = FALSE,
    theme = "theme_ggforge",
    theme_args = list(),
    palette = "Paired",
    palcolor = NULL,
    alpha = 1,
    aspect.ratio = NULL,
    legend.position = "right",
    legend.direction = "vertical",
    add_point = FALSE,
    pt_color = "grey30",
    pt_size = NULL,
    pt_alpha = 1,
    jitter_width = NULL,
    jitter_height = 0,
    stack = FALSE,
    y_max = NULL,
    y_min = NULL,
    add_trend = FALSE,
    trend_color = NULL,
    trend_linewidth = 1,
    trend_ptsize = 2,
    add_stat = NULL,
    stat_name = NULL,
    stat_color = "black",
    stat_size = 1,
    stat_stroke = 1,
    stat_shape = 25,
    add_bg = FALSE,
    bg_palette = "stripe",
    bg_palcolor = NULL,
    bg_alpha = 0.2,
    add_line = NULL,
    line_color = "red2",
    line_width = 0.6,
    line_type = 2,
    highlight = NULL,
    highlight_color = "red2",
    highlight_size = 1,
    highlight_alpha = 1,
    comparisons = NULL,
    ref_group = NULL,
    pairwise_method = "wilcox.test",
    multiplegroup_comparisons = FALSE,
    multiple_method = "kruskal.test",
    sig_label = "p.format",
    sig_labelsize = 3.5,
```

```
  hide_ns = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column for x-axis (discrete). Can be a single column name or multiple columns that will be concatenated.
y	Column for y-axis (numeric). The response variable.
in_form	Input data form: "long" (default) or "wide"
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
symnum_args	Symbolic number coding arguments for significance
sort_x	Sort x-axis values: "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median"
flip	Logical; flip coordinates to create horizontal plots
keep_empty	Logical; keep empty factor levels on x-axis
group_by	Column for grouping (creates dodged/side-by-side plots)
group_by_sep	Separator when concatenating multiple group_by columns
group_name	Legend name for groups
paired_by	Column identifying paired observations (for paired tests)
x_text_angle	Angle for x-axis text labels
step_increase	Step increase for comparison brackets
fill_mode	Fill coloring mode: "dodge", "x", "mean", or "median"
fill_reverse	Logical; reverse gradient fills
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name

palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
add_point	Logical; add jittered data points
pt_color	Point color (default: "grey30")
pt_size	Point size (auto-calculated if NULL)
pt_alpha	Point transparency (0-1)
jitter_width	Jitter width for points
jitter_height	Jitter height for points
stack	Logical; stack facets vertically/horizontally
y_max	Y-axis maximum (numeric or "qXX" for quantile)
y_min	Y-axis minimum (numeric or "qXX" for quantile)
add_trend	Logical; add trend line connecting medians
trend_color	Trend line color
trend_linewidth	Trend line width
trend_ptsize	Trend point size
add_stat	Function to add stat summary (e.g., mean)
stat_name	Stat legend name
stat_color	Stat point color
stat_size	Stat point size
stat_stroke	Stat point stroke width
stat_shape	Stat point shape
add_bg	Logical; add alternating background
bg_palette	Background color palette
bg_palcolor	Background custom colors
bg_alpha	Background transparency
add_line	Numeric; add horizontal reference line at this value
line_color	Reference line color
line_width	Reference line width
line_type	Reference line type
highlight	Points to highlight (logical, indices, or expression)
highlight_color	Highlight color
highlight_size	Highlight size
highlight_alpha	Highlight transparency
comparisons	Pairwise comparisons (list of pairs or TRUE for all)

ref_group	Reference group for comparisons
pairwise_method	Statistical method for pairwise comparisons
multiplegroup_comparisons	Logical; perform multiple group comparisons
multiple_method	Statistical method for multiple comparisons
sig_label	Significance label format: "p.format" or "p.signif"
sig_labelsize	Significance label font size
hide_ns	Logical; hide non-significant comparisons
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")

Value

A ggplot object or combined plots (patchwork)

Examples

```
# ============================================================
# Basic Examples
# ============================================================

# Create sample data
set.seed(123)
data <- data.frame(
  group = rep(LETTERS[1:4], each = 25),
  value = c(
    rnorm(25, 10, 2), rnorm(25, 12, 2),
    rnorm(25, 11, 3), rnorm(25, 14, 2)
  ),
```

```
  treatment = rep(c("Control", "Treatment"), 50),
  gender = sample(c("Male", "Female"), 100, replace = TRUE)
)

# Simple box plot
BoxPlot(data, x = "group", y = "value")

# Box plot with custom labels
BoxPlot(data,
  x = "group", y = "value",
  title = "Distribution by Group",
  xlab = "Experimental Group",
  ylab = "Measurement Value"
)

# ============================================================
# Grouped Box Plots
# ============================================================

# Side-by-side (dodged) box plots by treatment
BoxPlot(data, x = "group", y = "value", group_by = "treatment")

# With custom group legend name
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  group_name = "Treatment Group"
)

# With custom color palette
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  palette = "Set1"
)

# ============================================================
# Adding Data Points
# ============================================================

# Add jittered points to show individual observations
BoxPlot(data, x = "group", y = "value", add_point = TRUE)

# Customize point appearance
BoxPlot(data,
  x = "group", y = "value",
  add_point = TRUE,
  pt_color = "darkblue",
  pt_size = 1.5,
  pt_alpha = 0.6
)

# With grouped data
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  add_point = TRUE,
```

```
  pt_alpha = 0.5
)

# ============================================================
# Highlighting Specific Points
# ============================================================

# Highlight outliers or specific observations
BoxPlot(data,
  x = "group", y = "value",
  add_point = TRUE,
  highlight = "value > 15",
  highlight_color = "red",
  highlight_size = 3
)

# Highlight by row indices
BoxPlot(data,
  x = "group", y = "value",
  add_point = TRUE,
  highlight = c(1, 5, 10, 15),
  highlight_color = "orange"
)

# ============================================================
# Statistical Comparisons
# ============================================================

# Compare specific pairs
BoxPlot(data,
  x = "group", y = "value",
  comparisons = list(c("A", "B"), c("A", "D"))
)

# Compare all pairs
BoxPlot(data,
  x = "group", y = "value",
  comparisons = TRUE
)

# Use t-test instead of Wilcoxon
BoxPlot(data,
  x = "group", y = "value",
  comparisons = list(c("A", "D")),
  pairwise_method = "t.test"
)

# Show significance symbols instead of p-values
BoxPlot(data,
  x = "group", y = "value",
  comparisons = TRUE,
  sig_label = "p.signif"
)

# Hide non-significant comparisons
BoxPlot(data,
  x = "group", y = "value",
```

```
  comparisons = TRUE,
  hide_ns = TRUE
)

# With grouped data - compare groups within each x category
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  comparisons = TRUE
)

# Multiple group comparison (Kruskal-Wallis)
BoxPlot(data,
  x = "group", y = "value",
  multiplegroup_comparisons = TRUE
)

# ============================================================
# Paired Data Analysis
# ============================================================

# Create paired data (before/after measurements)
paired_data <- data.frame(
  time = factor(rep(c("Before", "After"), each = 20)),
  subject = factor(rep(1:20, 2)),
  score = c(rnorm(20, 50, 10), rnorm(20, 55, 10))
)

# Paired box plot with connecting lines
BoxPlot(paired_data,
  x = "time", y = "score",
  paired_by = "subject",
  add_point = TRUE
)

# With paired statistical test
BoxPlot(paired_data,
  x = "time", y = "score",
  paired_by = "subject",
  comparisons = list(c("Before", "After")),
  pairwise_method = "t.test"
)

# ============================================================
# Sorting and Orientation
# ============================================================

# Sort by mean value (ascending)
BoxPlot(data, x = "group", y = "value", sort_x = "mean_asc")

# Sort by median value (descending)
BoxPlot(data, x = "group", y = "value", sort_x = "median_desc")

# Horizontal box plot
BoxPlot(data, x = "group", y = "value", flip = TRUE)

# ============================================================
```

```
# Axis Customization
# ============================================================

# Set y-axis limits
BoxPlot(data,
  x = "group", y = "value",
  y_min = 5, y_max = 20
)

# Use quantiles for y-axis limits (exclude extreme values)
BoxPlot(data,
  x = "group", y = "value",
  y_min = "q5", y_max = "q95"
)

# Log-transform y-axis
pos_data <- data
pos_data$value <- abs(pos_data$value) + 1
BoxPlot(pos_data,
  x = "group", y = "value",
  y_trans = "log10"
)

# ============================================================
# Visual Enhancements
# ============================================================

# Add trend line connecting medians
BoxPlot(data, x = "group", y = "value", add_trend = TRUE)

# Add reference line
BoxPlot(data,
  x = "group", y = "value",
  add_line = 12,
  line_color = "red",
  line_type = 2
)

# Add alternating background
BoxPlot(data,
  x = "group", y = "value",
  add_bg = TRUE,
  bg_alpha = 0.1
)

# Add mean indicator
BoxPlot(data,
  x = "group", y = "value",
  add_stat = mean,
  stat_name = "Mean",
  stat_color = "red",
  stat_shape = 18
)

# ============================================================
# Faceting
# ============================================================
```

```
# Facet by another variable
BoxPlot(data,
  x = "group", y = "value",
  facet_by = "gender"
)

# Control facet layout
BoxPlot(data,
  x = "group", y = "value",
  facet_by = "gender",
  facet_ncol = 2
)

# Free y-axis scales per facet
BoxPlot(data,
  x = "group", y = "value",
  facet_by = "gender",
  facet_scales = "free_y"
)

# ============================================================
# Fill Modes
# ============================================================

# Fill by x-axis category (default when no group_by)
BoxPlot(data, x = "group", y = "value", fill_mode = "x")

# Fill by mean value (gradient)
BoxPlot(data,
  x = "group", y = "value",
  fill_mode = "mean",
  palette = "RdYlBu"
)

# Fill by median value (gradient)
BoxPlot(data,
  x = "group", y = "value",
  fill_mode = "median",
  palette = "viridis"
)

# ============================================================
# Wide Format Data
# ============================================================

# Wide format: each column is a group
wide_data <- data.frame(
  GroupA = rnorm(30, 10, 2),
  GroupB = rnorm(30, 12, 2),
  GroupC = rnorm(30, 11, 3)
)

BoxPlot(wide_data,
  x = c("GroupA", "GroupB", "GroupC"),
  in_form = "wide"
)
```

```
# ============================================================
# Splitting into Multiple Plots
# ============================================================

# Create separate plots by a variable
BoxPlot(data,
  x = "group", y = "value",
  split_by = "gender",
  combine = TRUE,
  ncol = 2
)

# ============================================================
# Theme and Style Customization
# ============================================================

# Custom color palette
BoxPlot(data,
  x = "group", y = "value",
  palette = "Dark2"
)

# Custom colors
BoxPlot(data,
  x = "group", y = "value",
  palcolor = c(
    "A" = "#E41A1C", "B" = "#377EB8",
    "C" = "#4DAF4A", "D" = "#984EA3"
  )
)

# Legend position
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  legend.position = "bottom",
  legend.direction = "horizontal"
)

# Hide legend
BoxPlot(data,
  x = "group", y = "value",
  group_by = "treatment",
  legend.position = "none"
)
```

`ChordPlot` *Chord / Circos Plot*

Description

Creates a chord diagram to visualize relationships between two categorical variables using the circlize package. Chord diagrams are useful for displaying flows or connections between entities, with link width proportional to relationship strength.

CircosPlot is an alias of ChordPlot.

Usage

```
ChordPlot(
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)

CircosPlot(
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
```

```
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
y	Column name for relationship values. If NULL, counts relationships
from	Column name(s) for source nodes. Multiple columns will be concatenated
from_sep	Separator for concatenating multiple from columns (default: "_")
to	Column name(s) for target nodes. Multiple columns will be concatenated
to_sep	Separator for concatenating multiple to columns (default: "_")
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
flip	Logical. Whether to flip source and target nodes (default: FALSE)
links_color	Character. Color links by source ("from") or target ("to") nodes
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
labels_rot	Logical. Whether to rotate labels by 90 degrees (default: FALSE)
title	Plot title
subtitle	Plot subtitle
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
...	Additional arguments passed to circlize functions

Value

If `combine = TRUE`, returns a combined patchwork object. If `combine = FALSE`, returns a list of individual chord diagrams.

Examples

```
set.seed(8525)
data <- data.frame(
  nodes1 = sample(c("Source1", "Source2", "Source3"), 10, replace = TRUE),
  nodes2 = sample(letters[1:3], 10, replace = TRUE),
  y = sample(1:5, 10, replace = TRUE)
)

# Basic chord plot
ChordPlot(data, from = "nodes1", to = "nodes2")

# With rotated labels and colored by target
ChordPlot(data,
  from = "nodes1", to = "nodes2",
  links_color = "to", labels_rot = TRUE
)

# With values
ChordPlot(data, from = "nodes1", to = "nodes2", y = "y")

# Split by variable
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y")

# With custom palettes per split
ChordPlot(data,
  from = "nodes1", to = "nodes2", split_by = "y",
  palette = c("1" = "Reds", "2" = "Blues", "3" = "Greens")
)

# Flip source and target
ChordPlot(data, from = "nodes1", to = "nodes2", flip = TRUE)
```

ClustreePlot *Clustree Plot*

Description

Visualizes clusterings at different resolutions using a tree structure. This function wraps the clustree package with consistent ggforge styling.

Usage

```
ClustreePlot(
  data,
  prefix,
  flip = FALSE,
  split_by = NULL,
```

```
  split_by_sep = "_",
  palette = "Paired",
  palcolor = NULL,
  edge_palette = "Spectral",
  edge_palcolor = NULL,
  alpha = 0.85,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(0.1, 0.1),
  theme = "theme_ggforge",
  theme_args = list(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
prefix	A character string of the prefix of the columns to plot. The columns with the prefix will be used to plot the tree.
flip	A logical value to flip the tree (horizontal layout).
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
palette	Color palette name
palcolor	Custom colors for palette
edge_palette	A character string of the palette name to color the edges.
edge_palcolor	A character vector of colors to color the edges.
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle

xlab	X-axis label
ylab	Y-axis label
expand	Expansion values for plot limits (CSS-like notation).
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
...	Other arguments passed to clustree::clustree.

Value

A ggplot object or combined plots (if split_by is used)

Examples

```
set.seed(8525)
N <- 100
data <- data.frame(
  p.0.4 = sample(LETTERS[1:5], N, replace = TRUE),
  p.0.5 = sample(LETTERS[1:6], N, replace = TRUE),
  p.0.6 = sample(LETTERS[1:7], N, replace = TRUE),
  p.0.7 = sample(LETTERS[1:8], N, replace = TRUE),
  p.0.8 = sample(LETTERS[1:9], N, replace = TRUE),
  p.0.9 = sample(LETTERS[1:10], N, replace = TRUE),
  p.1 = sample(LETTERS[1:30], N, replace = TRUE),
  split = sample(1:2, N, replace = TRUE)
)

ClustreePlot(data, prefix = "p")
ClustreePlot(data, prefix = "p", flip = TRUE)
ClustreePlot(data, prefix = "p", split_by = "split")
ClustreePlot(data,
  prefix = "p", split_by = "split",
  palette = c("1" = "Set1", "2" = "Paired")
)
```

CorPlot *Correlation Plot*

Description

Generate scatter correlation plot for two variables with optional linear regression line, annotations, and highlighting.

Usage

```
CorPlot(
  data,
  x,
  y,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  pt_size = 2,
  pt_shape = 16,
  raster = FALSE,
  alpha = 1,
  raster_dpi = c(512, 512),
  highlight = NULL,
  highlight_color = "black",
  highlight_size = 1,
  highlight_alpha = 1,
  highlight_stroke = 0.8,
  anno_items = c("n", "p", "pearson"),
  anno_size = 3.5,
  anno_fg = "black",
  anno_bg = "white",
  anno_bg_r = 0.1,
  anno_position = "auto",
  add_smooth = TRUE,
  smooth_color = "red2",
  smooth_width = 1.5,
  smooth_se = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
```

```
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for x-axis variable
y	Column name for y-axis variable
group_by	Column name(s) for grouping data
group_by_sep	Separator when concatenating multiple group_by columns
group_name	Name for the group legend
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
pt_size	Size of the points
pt_shape	Shape of the points (0-25)
raster	Whether to use raster graphics (faster for large datasets)
alpha	Transparency level (0-1)
raster_dpi	DPI for raster graphics as c(width, height)
highlight	Items to highlight. Can be: • A vector of row indices • A vector of rownames • An expression to filter (e.g., "Species == 'setosa'")
highlight_color	Color for highlighted points
highlight_size	Size for highlighted points
highlight_alpha	Alpha for highlighted points
highlight_stroke	Stroke width for highlighted points
anno_items	Annotation items to display. Options: "eq", "r2", "p", "spearman", "pearson", "kendall", "n"
anno_size	Size of annotation text
anno_fg	Foreground color of annotation text

anno_bg	Background color of annotation text
anno_bg_r	Radius of annotation background
anno_position	Position of annotations. Options: "auto", "topleft", "topright", "bottomleft", "bottomright" (or shortcuts: "tl", "tr", "bl", "br")
add_smooth	Whether to add linear regression line
smooth_color	Color of regression line
smooth_width	Width of regression line
smooth_se	Whether to show standard error band
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or list/combined plots if split_by is used

Examples

```
# Basic correlation plot with grouping
data(iris)
CorPlot(iris, x = "Sepal.Length", y = "Sepal.Width", group_by = "Species")

# With custom annotations and positioning
CorPlot(iris,
  x = "Sepal.Length", y = "Sepal.Width",
  group_by = "Species",
  anno_items = c("n", "eq", "r2", "pearson"),
  anno_position = "bottomright"
)

# With highlighting specific points
CorPlot(iris,
  x = "Sepal.Length", y = "Sepal.Width",
  group_by = "Species",
  highlight = 'Species == "setosa"',
  highlight_color = "red",
  highlight_size = 3,
  highlight_stroke = 1.5
)

# With faceting by groups
CorPlot(iris,
  x = "Sepal.Length", y = "Sepal.Width",
  facet_by = "Species",
  facet_scales = "free",
  add_smooth = TRUE,
  smooth_color = "blue"
)

# With splitting and custom palettes
CorPlot(iris,
  x = "Sepal.Length", y = "Sepal.Width",
  split_by = "Species",
  palette = c(setosa = "Set1", versicolor = "Dark2", virginica = "Paired"),
  combine = TRUE
)

# For large datasets, use raster mode
## Not run:
CorPlot(large_data,
  x = "x_var", y = "y_var",
  raster = TRUE,
  raster_dpi = c(1024, 1024),
  pt_size = 1
)

## End(Not run)
```

DensityPlot *Density Plot*

Description

Creates density plots to illustrate the distribution of continuous data. Supports grouping, faceting, and splitting into multiple plots.

Usage

```
DensityPlot(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  position = "identity",
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  theme = "theme_ggforge",
  theme_args = list(),
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  legend.position = waiver(),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for values (numeric expected)
group_by	Column to group the data
group_by_sep	Separator for concatenating multiple group_by columns
group_name	Legend title for group_by
xtrans	Transformation for x-axis (e.g., "identity", "log10")
ytrans	Transformation for y-axis
split_by	Column to split data into multiple plots
split_by_sep	Separator for concatenating multiple split_by columns
flip	Whether to flip the plot
position	Position adjustment for overlapping groups
palette	Palette name
palcolor	Custom colors
alpha	Transparency for density curves
theme	Theme name or function
theme_args	Arguments passed to theme function
add_bars	Whether to add data distribution lines at bottom
bar_height	Height of distribution bars (as fraction of max)
bar_alpha	Alpha for distribution bars
bar_width	Width of distribution bars
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
expand	Axis expansion
facet_by	Columns for faceting
facet_scales	Facet scales type
facet_ncol	Number of facet columns
facet_nrow	Number of facet rows
facet_byrow	Fill facets by row
legend.position	Legend position
legend.direction	Legend direction
seed	Random seed
combine	Whether to combine multiple plots
nrow	Number of rows for combined plots
ncol	Number of columns for combined plots
byrow	Fill combined plots by row
axes	Axis handling for combined plots
axis_titles	Axis title handling for combined plots
guides	Guide handling for combined plots
design	Custom design for combined plots
...	Additional arguments passed to geom_density

Value

A ggplot object, combined plot, or list of plots

Examples

```
set.seed(8525)
data <- data.frame(
  x = c(rnorm(500, -1), rnorm(500, 1)),
  group = rep(c("A", "B"), each = 500),
  facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

DensityPlot(data, x = "x")
DensityPlot(data, x = "x", group_by = "group")
DensityPlot(data, x = "x", group_by = "group", facet_by = "facet")
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE)
```

DimPlot *Dimension Reduction Plot*

Description

Visualizes dimension reduction data (UMAP, t-SNE, PCA, etc.) as scatter plots. Supports both categorical grouping and continuous feature plotting with extensive customization options.

Usage

```
DimPlot(
  data,
  dims = 1:2,
  group_by = NULL,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  label = FALSE,
  label_insitu = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  show_stat = !identical(theme, "theme_blank"),
  order = c("as-is", "reverse", "high-top", "low-top", "random"),
  highlight = NULL,
  highlight_alpha = 1,
```

```
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = grid::unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = TRUE,
  hex_bins = 50,
  hex_binwidth = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge_grid",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot

dims	Column names or indices for x and y axes. Default is first two columns.
group_by	Column name for categorical grouping. Creates discrete color mapping. If multiple columns provided, they will be concatenated with group_by_sep.
group_by_sep	Separator for concatenating multiple group_by columns.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
pt_size	Point size. If NULL, calculated based on number of data points.
pt_alpha	Point transparency (0-1).
bg_color	Color for NA/background points.
label	Whether to show group labels at median positions.
label_insitu	Whether to use actual group names as labels (vs. numbers).
label_size	Size of labels.
label_fg, label_bg	Foreground and background colors for labels.
label_bg_r	Background radius for labels.
label_repel	Whether to repel overlapping labels.
label_repulsion	Repulsion force for labels.
label_pt_size, label_pt_color	Point size and color for label anchors.
label_segment_color	Color of segments connecting labels to points.
show_stat	Whether to show sample count in subtitle.
order	How to order points: "as-is", "reverse", "high-top", "low-top", "random". Affects drawing order (what's on top).
highlight	Row names/indices to highlight, or logical expression as string.
highlight_alpha, highlight_size, highlight_color, highlight_stroke	Styling for highlighted points.
add_mark	Whether to add marks (hulls/ellipses) around groups.
mark_type	Type of mark: "hull", "ellipse", "rect", "circle".
mark_expand, mark_alpha, mark_linetype	Mark styling parameters.
add_density	Whether to add 2D density contours/fill.
density_color, density_filled	Density styling.
density_filled_palette, density_filled_palcolor	Palette for filled density.
raster	Whether to rasterize points (useful for large datasets).
raster_dpi	DPI for rasterization.
hex	Whether to use hexagonal binning instead of points.
hex_linewidth, hex_count, hex_bins, hex_binwidth	Hex bin parameters.
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"

facet_nrow	Number of rows in facet layout
facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Keep empty factor levels
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or list/combined plots

Examples

```
## Not run:
# Basic usage
DimPlot(data, dims = c("UMAP_1", "UMAP_2"), group_by = "clusters")

# With faceting
DimPlot(data, dims = 1:2, group_by = "clusters", facet_by = "condition")

# With splitting
DimPlot(data, dims = 1:2, group_by = "clusters", split_by = "timepoint")

## End(Not run)
```

dim_example *Example Dimensionality Reduction Data*

Description

Example data for dimensionality reduction plots (e.g., UMAP, t-SNE). Contains cell embeddings, velocity, and clustering information.

Usage

```
dim_example
```

Format

A data frame with:

Embeddings First 4 columns: x, y coordinates and velocity

clusters Cell cluster assignments

Additional metadata Additional grouping variables

DotPlot *Dot Plot / Scatter Plot*

Description

Creates a dot plot where both X and Y axes can be numeric or categorical. When both are numeric, the plot functions as a scatter plot. Supports sizing dots by a numeric column and filling by another numeric column.

Usage

```
DotPlot(
  data,
  x,
  y,
  x_sep = "_",
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_reverse = FALSE,
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  add_bg = FALSE,
  bg_palette = "stripe",
```

```
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  bg_direction = c("vertical", "horizontal", "v", "h"),
  lollipop = FALSE,
  theme = "theme_ggforge_grid",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	A character string specifying the column to use for the x-axis. Can be numeric or factor/character. When multiple columns are provided, they will be concatenated with x_sep.
y	A character string specifying the column to use for the y-axis. Can be numeric or factor/character. When multiple columns are provided, they will be concatenated with y_sep.
x_sep	A character string to concatenate multiple columns in x.
y_sep	A character string to concatenate multiple columns in y.
flip	Whether to flip the x and y axes.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns

size_by	Which column to use as the size of the dots (numeric column). If not provided, the size will be the count of instances for each (x, y) pair. Can also be a single numeric value to specify a fixed size.
fill_by	Which column to use to fill the dots (numeric column). If not provided, all dots will be filled with the middle color of the palette.
fill_cutoff	A numeric value specifying the cutoff for the fill column. Values below (or above if `fill_reverse = TRUE`) will be shown in grey.
fill_reverse	Whether to reverse the fill direction. If FALSE (default), values < cutoff are grey. If TRUE, values > cutoff are grey.
size_name	A character string to name the size legend.
fill_name	A character string to name the fill legend.
fill_cutoff_name	A character string to name the fill cutoff legend.
add_bg	Whether to add a background color to the plot.
bg_palette	Palette for the background color.
bg_palcolor	Custom colors for the background.
bg_alpha	Alpha value for the background color.
bg_direction	Direction for background stripes ("vertical" or "horizontal").
lollipop	Whether to create a lollipop plot (requires numeric x and factor y).
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
x_text_angle	Angle for x-axis text.
seed	Random seed for reproducibility
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Whether to keep empty factor levels.
combine	Whether to combine split plots into one

nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
mtcars <- datasets::mtcars
mtcars$carb <- factor(mtcars$carb)
mtcars$gear <- factor(mtcars$gear)
DotPlot(mtcars,
  x = "carb", y = "gear", size_by = "wt",
  fill_by = "mpg", fill_cutoff = 18
)
DotPlot(mtcars,
  x = "carb", y = "gear", size_by = "wt",
  fill_by = "mpg", fill_cutoff = 18, add_bg = TRUE
)
# Scatter plot (both axes numeric)
DotPlot(mtcars,
  x = "qsec", y = "drat", size_by = "wt",
  fill_by = "mpg", fill_cutoff = 18
)
```

EnrichMap *Enrichment Map*

Description

Creates enrichment map visualizations showing clustering of enriched terms based on gene overlap. Terms are nodes, and edges represent shared genes. Nodes are clustered to reveal functional modules.

Usage

```
EnrichMap(
  data,
  in_form = c("auto", "clusterProfiler", "clusterprofiler", "enrichr"),
  split_by = NULL,
  split_by_sep = "_",
  top_term = 10,
  metric = "p.adjust",
```

```
  layout = "fr",
  minchar = 2,
  cluster = "fast_greedy",
  show_keyword = FALSE,
  nlabel = 4,
  character_width = 50,
  mark = "ellipse",
  label = c("term", "feature"),
  labelsize = 5,
  expand = c(0.4, 0.4),
  words_excluded = NULL,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

`data` A data frame containing the data to be plotted. It should be in the format of clusterProfiler enrichment result, which includes the columns: ID, Description, GeneRatio, BgRatio, pvalue, p.adjust, qvalue, geneID and Count.

- The `ID`, `qvalue` and `Count` columns are optional.
- The `Description` is the description of the term.
- The `GeneRatio` is the number of genes in the term divided by the total number of genes in the input list.
- The `BgRatio` is the number of genes in the term divided by the total number of genes in the background list (all terms).
- The `Count` column, if given, should be the same as the first number in GeneRatio.

If you have enrichment results from multiple databases, you can combine them into one data frame and add a column (e.g. Database) to indicate the database. You can plot them in a single plot using the `split_by` argument (e.g. `split_by = "Database"`).

in_form	A character string specifying the input format. Either "auto", "clusterProfiler", "clusterprofiler" or "enrichr". The default is "auto", which will try to infer the input format.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
top_term	An integer specifying the number of top terms to show.
metric	A character string specifying the metric to use for the size of the nodes. It is also used to order the terms when selected the top terms. Either "pvalue" or "p.adjust". The default is "p.adjust".
layout	A character string specifying the layout of the graph. Either "circle", "tree", "grid" or other layout functions in igraph.
minchar	An integer specifying the minimum number of characters to show in the keyword.
cluster	A character string specifying the clustering method. Either "fast_greedy", "walktrap", "edge_betweenness", "infomap" or other clustering functions in igraph.
show_keyword	A logical value specifying whether to show the keyword instead of Description/Term in the plot.
nlabel	An integer specifying the number of labels to show in each cluster.
character_width	The width of the characters used to wrap the keyword.
mark	A character string specifying the mark to use for the nodes. Either "ellipse", "rect", "circle", "text" or other mark functions in ggforce.
label	A character string specifying the label to show in the legend. Either "term" or "feature". The default is "term".
labelsize	A numeric value specifying the size of the label.
expand	Expansion values for plot axes (CSS-like padding)
words_excluded	A character vector specifying the words to exclude in the keyword.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots

ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data(enrich_example)
EnrichMap(enrich_example)
EnrichMap(enrich_example, label = "feature")
EnrichMap(enrich_example, show_keyword = TRUE, label = "term")
EnrichMap(enrich_example, show_keyword = TRUE, label = "feature")

data(enrich_multidb_example)
EnrichMap(enrich_multidb_example, split_by = "Database")
EnrichMap(enrich_multidb_example,
  split_by = "Database",
  palette = list(DB1 = "Paired", DB2 = "Set1")
)
```

EnrichNetwork *Enrichment Network*

Description

Creates enrichment network visualizations showing connections between enriched terms and their associated genes. Terms and genes are both shown as nodes, with edges connecting terms to their genes.

Usage

```
EnrichNetwork(
  data,
  in_form = c("auto", "clusterProfiler", "clusterprofiler", "enrichr"),
  split_by = NULL,
  split_by_sep = "_",
  top_term = 10,
  metric = "p.adjust",
  character_width = 50,
  layout = "fr",
  layoutadjust = TRUE,
  adjscale = 60,
  adjiter = 100,
```

```
  blendmode = "blend",
  labelsize = 5,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to be plotted. It should be in the format of clusterProfiler enrichment result, which includes the columns: ID, Description, GeneRatio, BgRatio, pvalue, p.adjust, qvalue, geneID and Count. • The `ID`, `qvalue` and `Count` columns are optional. • The `Description` is the description of the term. • The `GeneRatio` is the number of genes in the term divided by the total number of genes in the input list. • The `BgRatio` is the number of genes in the term divided by the total number of genes in the background list (all terms). • The `Count` column, if given, should be the same as the first number in GeneRatio. If you have enrichment results from multiple databases, you can combine them into one data frame and add a column (e.g. Database) to indicate the database. You can plot them in a single plot using the `split_by` argument (e.g. `split_by = "Database"`).
in_form	A character string specifying the input format. Either "auto", "clusterProfiler", "clusterprofiler" or "enrichr". The default is "auto", which will try to infer the input format.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
top_term	An integer specifying the number of top terms to show.

metric	A character string specifying the metric to use for the size of the nodes. It is also used to order the terms when selected the top terms. Either "pvalue" or "p.adjust". The default is "p.adjust".
character_width	The width of the characters used to wrap the keyword.
layout	A character string specifying the layout of the graph. Either "circle", "tree", "grid" or other layout functions in igraph.
layoutadjust	A logical value specifying whether to adjust the layout of the network.
adjscale	A numeric value specifying the scale of the adjustment.
adjiter	A numeric value specifying the number of iterations for the adjustment.
blendmode	A character string specifying the blend mode of the colors. Either "blend", "average", "multiply" and "screen".
labelsize	A numeric value specifying the size of the label.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data(enrich_example)
EnrichNetwork(enrich_example, top_term = 5)
```

enrich_example *Example Enrichment Analysis Data*

Description

Example data from clusterProfiler enrichment analysis. Used for demonstrating enrichment plot functions.

Usage

```
enrich_example
```

Format

A data frame with enrichment results

enrich_multidb_example *Example Multi-Database Enrichment Data*

Description

Example enrichment data from multiple databases.

Usage

```
enrich_multidb_example
```

Format

A data frame with enrichment results from multiple databases

FeatureDimPlot *Feature Dimension Reduction Plot*

Description

Visualizes continuous features (gene expression, etc.) on dimension reduction plots

Usage

```
FeatureDimPlot(
  data,
  dims = 1:2,
  features,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  bg_cutoff = NULL,
  color_name = "",
  split_by = NULL,
  split_by_sep = "_",
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  show_stat = !identical(theme, "theme_blank"),
  order = c("as-is", "reverse", "high-top", "low-top", "random"),
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
  hex_binwidth = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge_grid",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
```

```
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
dims	Column names or indices for x and y axes. Default is first two columns.
features	Column name(s) of continuous features to plot. If multiple features provided, creates separate plots for each.
lower_quantile, upper_quantile	Quantiles for color scale limits (0-1).
lower_cutoff, upper_cutoff	Explicit cutoff values for color scale.
bg_cutoff	Values below this threshold are set to NA (background color).
color_name	Legend title for the color scale.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
pt_size	Point size. If NULL, calculated based on number of data points.
pt_alpha	Point transparency (0-1).
bg_color	Color for NA/background points.
label	Whether to show group labels at median positions.
label_size	Size of labels.
label_fg, label_bg	Foreground and background colors for labels.
label_bg_r	Background radius for labels.
show_stat	Whether to show sample count in subtitle.
order	How to order points: "as-is", "reverse", "high-top", "low-top", "random". Affects drawing order (what's on top).
highlight	Row names/indices to highlight, or logical expression as string.
highlight_alpha, highlight_size, highlight_color, highlight_stroke	Styling for highlighted points.
add_density	Whether to add 2D density contours/fill.

density_color, density_filled
: Density styling.

density_filled_palette, density_filled_palcolor
: Palette for filled density.

raster
: Whether to rasterize points (useful for large datasets).

raster_dpi
: DPI for rasterization.

hex
: Whether to use hexagonal binning instead of points.

hex_linewidth, hex_count, hex_bins, hex_binwidth
: Hex bin parameters.

facet_by
: Column name(s) for faceting the plot

facet_scales
: Scales for facets: "fixed", "free", "free_x", "free_y"

facet_nrow
: Number of rows in facet layout

facet_ncol
: Number of columns in facet layout

facet_byrow
: Fill facets by row (TRUE) or column (FALSE)

theme
: Theme name (string) or theme function

theme_args
: List of arguments passed to theme function

palette
: Color palette name

palcolor
: Custom colors for palette

alpha
: Transparency level (0-1)

aspect.ratio
: Aspect ratio of plot panel

legend.position
: Legend position: "none", "left", "right", "bottom", "top"

legend.direction
: Legend direction: "horizontal" or "vertical"

title
: Plot title

subtitle
: Plot subtitle

xlab
: X-axis label

ylab
: Y-axis label

seed
: Random seed for reproducibility

combine
: Whether to combine split plots into one

nrow
: Number of rows when combining plots

ncol
: Number of columns when combining plots

byrow
: Fill combined plots by row

axes
: How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")

axis_titles
: How to handle axis titles in combined plots

guides
: How to handle guides in combined plots ("collect", "keep", "auto")

design
: Custom layout design for combined plots

Value

A ggplot object or list/combined plots

Examples

```
## Not run:
# Single feature
FeatureDimPlot(data, features = "gene1", dims = 1:2)

# Multiple features
FeatureDimPlot(data, features = c("gene1", "gene2"), dims = 1:2)

# With splits
FeatureDimPlot(data, features = "gene1", split_by = "condition")

## End(Not run)
```

`get_palette` *Get colors from palette*

Description

Main function for retrieving colors from palettes with intelligent handling of discrete and continuous data.

Usage

```
get_palette(
  x,
  n = 100,
  palette = "Paired",
  palcolor = NULL,
  type = "auto",
  keep_names = TRUE,
  alpha = 1,
  matched = FALSE,
  reverse = FALSE,
  NA_keep = FALSE,
  NA_color = "grey80",
  transparent = TRUE
)
```

Arguments

`x`	Vector of values to map to colors (character, factor, or numeric)
`n`	Number of colors for continuous palettes
`palette`	Name of the palette to use
`palcolor`	Custom colors (overrides palette)
`type`	Type of palette: "auto", "discrete", or "continuous"
`keep_names`	Whether to keep names on color vector
`alpha`	Transparency level (0-1)
`matched`	Return colors matched to x values
`reverse`	Reverse the color order

NA_keep	Include color for NA values
NA_color	Color for NA values
transparent	Use true transparency vs color blending

Value

Named vector of colors

Examples

```
# Discrete palette
get_palette(c("A", "B", "C"), palette = "Paired")

# Continuous palette
get_palette(1:100, palette = "Spectral", type = "continuous")

# Custom colors
get_palette(c("A", "B", "C"), palcolor = c("red", "blue", "green"))
```

GSEAPlot *GSEA Plot*

Description

Creates detailed GSEA plots for one or more gene sets, showing the running enrichment score, gene positions, and ranked list metric for each set.

Usage

```
GSEAPlot(
  data,
  in_form = c("auto", "dose", "fgsea"),
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
  gs = NULL,
  sample_coregenes = FALSE,
  line_width = 1.5,
  line_alpha = 1,
  line_color = "#6BB82D",
  n_coregenes = 10,
  genes_label = NULL,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  label_size = 4,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
```

```
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  theme = "theme_ggforge",
  ...
)
```

Arguments

data	A data frame containing the data to plot
in_form	The format of the input data • fgsea: The input data is from the fgsea package. • dose: The input data is from the DOSE package. • auto: Automatically detect the format of the input data. When "leadingEdge" is in the input data, it will be treated as "fgsea"; otherwise, if "core_enrichment" is in the input data, it will be treated as "dose".
gene_ranks	A numeric vector of gene ranks with genes as names The gene ranks are used to plot the gene sets. If gene_ranks is a character vector starting with @, the gene ranks will be taken from the attribute of data.
gene_sets	A list of gene sets, typically from a record of a GMT file The names of the list should match the ID column of data. If gene_sets is a character vector starting with @, the gene sets will be taken from the attribute of data. The GSEA plots will be plotted for each gene set. So, the number of plots will be the number of gene sets. If you only want to plot a subset of gene sets, you can subset the gene_sets before passing it to this function.
gs	The names of the gene sets to plot If NULL, all gene sets in gene_sets will be plotted.
sample_coregenes	A logical value to sample the core genes from the core_enrichment; if FALSE, the first n_coregenes will be used
line_width	The width of the line in the running score plot
line_alpha	The alpha of the line in the running score plot
line_color	The color of the line in the running score plot
n_coregenes	The number of core genes to label
genes_label	The genes to label. If set, n_coregenes will be ignored
label_fg	The color of the label text
label_bg	The background color of the label
label_bg_r	The radius of the background color of the label
label_size	The size of the label text
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label

combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
theme	Theme name (string) or theme function

Value

A ggplot object (if single plot) or combined plot object

Examples

```
data(gsea_example)

# Plot single gene set
GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1])

# Plot multiple gene sets
GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1:4])

# Label core genes
GSEAPlot(
  gsea_example,
  gene_sets = attr(gsea_example, "gene_sets")[1],
  n_coregenes = 5
)

# Customize line appearance
GSEAPlot(
  gsea_example,
  gene_sets = attr(gsea_example, "gene_sets")[1],
  line_width = 2,
  line_color = "#FF6B6B"
)

# Return separate plots instead of combined
plots <- GSEAPlot(
  gsea_example,
  gene_sets = attr(gsea_example, "gene_sets")[1:3],
  combine = FALSE
)
```

GSEASummaryPlot *GSEA Summary Plot*

Description

Creates a summary visualization of GSEA (Gene Set Enrichment Analysis) results, showing normalized enrichment scores (NES) with integrated line plots for each term.

Usage

```
GSEASummaryPlot(
  data,
  in_form = c("auto", "dose", "fgsea"),
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
  top_term = 10,
  metric = "p.adjust",
  cutoff = 0.05,
  character_width = 50,
  line_plot_size = 0.25,
  metric_name = metric,
  nonsig_name = "Insignificant",
  linewidth = 0.2,
  line_by = c("prerank", "running_score"),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  alpha = 0.6,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  seed = 8525,
  ...
)
```

Arguments

data	A data frame of GSEA results For example, from `DOSE::gseDO()` or `fgsea::fgsea()`. Required columns are `ID`, `Description`, `NES`, `p.adjust`, `pvalue`. The `ID` column is used to match the gene sets.
in_form	The format of the input data • `fgsea`: The input data is from the `fgsea` package. • `dose`: The input data is from the `DOSE` package. • `auto`: Automatically detect the format of the input data. When "leadingEdge" is in the input data, it will be treated as "fgsea"; otherwise, if "core_enrichment" is in the input data, it will be treated as "dose".

gene_ranks — A numeric vector of gene ranks with genes as names The gene ranks are used to plot the gene sets. If gene_ranks is a character vector starting with @, the gene ranks will be taken from the attribute of data.

gene_sets — A list of gene sets, typically from a record of a GMT file The names of the list should match the ID column of data. If gene_sets is a character vector starting with @, the gene sets will be taken from the attribute of data.

top_term — An integer to select the top terms

metric — The metric to use for the significance of the terms Typically the column name of p values or adjusted p values. It is also used to select the top terms.

cutoff — The cutoff for the significance of the terms The terms will not be filtered with this cutoff; they are only filtered by the top_term ranked by the metric. The cutoff here is used to show the significance of the terms on the plot. For the terms that are not significant, the color will be grey.

character_width — The width of the characters in the y-axis

line_plot_size — The size of the line plots

metric_name — The name of the metric to show in the color bar

nonsig_name — The name of the legend for the nonsignificant terms

linewidth — The width of the lines in the line plots

line_by — The method to calculate the line plots.

- prerank: Use the gene ranks as heights to plot the line plots.
- running_score: Use the running score to plot the line plots.

title — Plot title

subtitle — Plot subtitle

xlab — X-axis label

ylab — Y-axis label

alpha — Transparency level (0-1)

aspect.ratio — Aspect ratio of plot panel

legend.position — Legend position: "none", "left", "right", "bottom", "top"

legend.direction — Legend direction: "horizontal" or "vertical"

theme — Theme name (string) or theme function

theme_args — List of arguments passed to theme function

palette — Color palette name

palcolor — Custom colors for palette

seed — Random seed for reproducibility

Value

A ggplot object with height and width attributes

Examples

```
data(gsea_example)

# Basic usage
GSEASummaryPlot(gsea_example)

# Use running score instead of prerank
GSEASummaryPlot(gsea_example, line_by = "running_score")

# Adjust significance cutoff
GSEASummaryPlot(gsea_example, cutoff = 0.01)

# Show more terms
GSEASummaryPlot(gsea_example, top_term = 15)

# Customize color palette
GSEASummaryPlot(gsea_example, palette = "RdYlBu")

# Adjust character width for long pathway names
GSEASummaryPlot(gsea_example, character_width = 70)
```

---

gsea_example *Example GSEA Results*

Description

Example Gene Set Enrichment Analysis results.

Usage

```
gsea_example
```

Format

A data frame with GSEA results

Heatmap *Heatmap*

Description

Heatmap is a popular way to visualize data in matrix format. It is widely used in biology to visualize gene expression data in microarray and RNA-seq data. The heatmap is a matrix where rows represent the samples and columns represent the features. The color of each cell represents the value of the feature in the sample. The color can be continuous or discrete. The heatmap can be split by the columns or rows to show the subgroups in the data. The heatmap can also be annotated by the columns or rows to show the additional information of the samples or features.

Usage

```
Heatmap(
  data,
  values_by = NULL,
  values_fill = NA,
  name = NULL,
  in_form = c("auto", "matrix", "wide-columns", "wide-rows", "long"),
  split_by = NULL,
  split_by_sep = "_",
  rows_by = NULL,
  rows_by_sep = "_",
  rows_split_by = NULL,
  rows_split_by_sep = "_",
  columns_by = NULL,
  columns_by_sep = "_",
  columns_split_by = NULL,
  columns_split_by_sep = "_",
  rows_data = NULL,
  columns_data = NULL,
  columns_name = NULL,
  columns_split_name = NULL,
  rows_name = NULL,
  rows_split_name = NULL,
  palette = "RdBu",
  palcolor = NULL,
  rows_palette = "Paired",
  rows_palcolor = NULL,
  rows_split_palette = "simspec",
  rows_split_palcolor = NULL,
  columns_palette = "Paired",
  columns_palcolor = NULL,
  columns_split_palette = "simspec",
  columns_split_palcolor = NULL,
  pie_size_name = "size",
  pie_size = NULL,
  pie_values = "length",
  pie_name = NULL,
  pie_group_by = NULL,
  pie_group_by_sep = "_",
  pie_palette = "Spectral",
  pie_palcolor = NULL,
  bars_sample = 100,
  label = identity,
  label_size = 10,
  violin_fill = NULL,
  boxplot_fill = NULL,
  dot_size = 8,
  dot_size_name = "size",
  legend_items = NULL,
  legend_discrete = FALSE,
  legend.position = "right",
  legend.direction = "vertical",
```

```
lower_quantile = 0,
upper_quantile = 0.99,
lower_cutoff = NULL,
upper_cutoff = NULL,
add_bg = FALSE,
bg_alpha = 0.5,
add_reticle = FALSE,
reticle_color = "grey",
column_name_annotation = TRUE,
column_name_legend = NULL,
row_name_annotation = TRUE,
row_name_legend = NULL,
cluster_columns = TRUE,
cluster_rows = TRUE,
show_row_names = !row_name_annotation,
show_column_names = !column_name_annotation,
border = TRUE,
title = NULL,
column_title = character(0),
row_title = character(0),
na_col = "grey85",
row_names_side = "right",
column_names_side = "bottom",
column_annotation = NULL,
column_annotation_side = "top",
column_annotation_palette = "Paired",
column_annotation_palcolor = NULL,
column_annotation_type = "auto",
column_annotation_params = list(),
column_annotation_agg = NULL,
row_annotation = NULL,
row_annotation_side = "left",
row_annotation_palette = "Paired",
row_annotation_palcolor = NULL,
row_annotation_type = "auto",
row_annotation_params = list(),
row_annotation_agg = NULL,
flip = FALSE,
alpha = 1,
seed = 8525,
layer_fun_callback = NULL,
cell_type = c("tile", "bars", "label", "dot", "violin", "boxplot", "pie"),
cell_agg = NULL,
combine = TRUE,
nrow = NULL,
ncol = NULL,
byrow = TRUE,
axes = NULL,
axis_titles = axes,
guides = NULL,
design = NULL,
...
```

)

Arguments

data	A data frame or matrix containing the data to be plotted. Based on the in_form, the data can have the following formats: • matrix: A matrix with rows and columns directly representing the heatmap. • long: A data frame in long format with columns for values, rows, and columns. • wide-rows: A data frame in wide format with columns for heatmap rows and values, and a single column for heatmap columns. • wide-columns: A data frame in wide format with columns for heatmap columns and values, and a single column for heatmap rows. • auto: Automatically inferred from the data format. When data is a matrix, in_form is set to "matrix". When columns_by has more than one column, in_form is set to "wide-columns". When rows_by has more than one column, in_form is set to "wide-rows". Otherwise, it is set to "long".
values_by	A character of column name in data that contains the values to be plotted. This is required when in_form is "long". For other formats, the values are pivoted into a column named by values_by.
values_fill	A value to fill in the missing values in the heatmap. When there is missing value in the data, the cluster_rows and cluster_columns will fail.
name	A character string to name the heatmap (will be used to rename values_by).
in_form	The format of the data. Can be one of "matrix", "long", "wide-rows", "wide-columns", or "auto". Defaults to "auto".
split_by	A character of column name in data that contains the split information to split into multiple heatmaps. This is used to create a list of heatmaps, one for each level of the split. Defaults to NULL, meaning no split.
split_by_sep	A character string to concat multiple columns in split_by.
rows_by	A vector of column names in data that contains the row information. This is used to create the rows of the heatmap. When in_form is "long" or "wide-columns", this is requied, and multiple columns can be specified, which will be concatenated by rows_by_sep into a single column.
rows_by_sep	A character string to concat multiple columns in rows_by.
rows_split_by	A character of column name in data that contains the split information for rows.
rows_split_by_sep	A character string to concat multiple columns in rows_split_by.
columns_by	A vector of column names in data that contains the column information. This is used to create the columns of the heatmap. When in_form is "long" or "wide-rows", this is required, and multiple columns can be specified, which will be concatenated by columns_by_sep into a single column.
columns_by_sep	A character string to concat multiple columns in columns_by.
columns_split_by	A character of column name in data that contains the split information for columns.
columns_split_by_sep	A character string to concat multiple columns in columns_split_by.

rows_data
: A data frame containing additional data for rows, which can be used to add annotations to the heatmap. It will be joined to the main data by rows_by and split_by if split_by exists in rows_data. This is useful for adding additional information to the rows of the heatmap.

columns_data
: A data frame containing additional data for columns, which can be used to add annotations to the heatmap. It will be joined to the main data by columns_by and split_by if split_by exists in columns_data. This is useful for adding additional information to the columns of the heatmap.

columns_name
: A character string to rename the column created by columns_by, which will be reflected in the name of the annotation or legend.

columns_split_name
: A character string to rename the column created by columns_split_by, which will be reflected in the name of the annotation or legend.

rows_name
: A character string to rename the column created by rows_by, which will be reflected in the name of the annotation or legend.

rows_split_name
: A character string to rename the column created by rows_split_by, which will be reflected in the name of the annotation or legend.

palette
: A character string specifying the palette of the heatmap cells.

palcolor
: A character vector of colors to override the palette of the heatmap cells.

rows_palette
: A character string specifying the palette of the row group annotation. The default is "Paired".

rows_palcolor
: A character vector of colors to override the palette of the row group annotation.

rows_split_palette
: A character string specifying the palette of the row split annotation. The default is "simspec".

rows_split_palcolor
: A character vector of colors to override the palette of the row split annotation.

columns_palette
: A character string specifying the palette of the column group annotation. The default is "Paired".

columns_palcolor
: A character vector of colors to override the palette of the column group annotation.

columns_split_palette
: A character string specifying the palette of the column split annotation. The default is "simspec".

columns_split_palcolor
: A character vector of colors to override the palette of the column split annotation.

pie_size_name
: A character string specifying the name of the legend for the pie size.

pie_size
: A numeric value or a function specifying the size of the pie chart. If it is a function, the function should take count as the argument and return the size.

pie_values
: A function or character that can be converted to a function by match.arg() to calculate the values for the pie chart. Default is "length". The function should take a vector of values as the argument and return a single value, for each group in pie_group_by.

pie_name
: A character string to rename the column created by pie_group_by, which will be reflected in the name of the annotation or legend.

pie_group_by
: A character of column name in data that contains the group information for pie charts. This is used to create pie charts in the heatmap when cell_type is "pie".

pie_group_by_sep
: A character string to concat multiple columns in pie_group_by.

pie_palette
: A character string specifying the palette of the pie chart.

pie_palcolor
: A character vector of colors to override the palette of the pie chart.

bars_sample
: An integer specifying the number of samples to draw the bars.

label
: A function to calculate the labels for the heatmap cells. It can take either 1, 3, or 5 arguments. The first argument is the aggregated values. If it takes 3 arguments, the second and third arguments are the row and column indices. If it takes 5 arguments, the second and third arguments are the row and column indices, the fourth and fifth arguments are the row and column names. The function should return a character vector of the same length as the aggregated values. If the function returns NA, no label will be shown for that cell. For the indices, if you have the same dimension of data (same order of rows and columns) as the heatmap, you need to use ComplexHeatmap::pindex() to get the correct values.

label_size
: A numeric value specifying the size of the labels when cell_type = "label".

violin_fill
: A character vector of colors to override the fill color of the violin plot. If NULL, the fill color will be the same as the annotion.

boxplot_fill
: A character vector of colors to override the fill color of the boxplot. If NULL, the fill color will be the same as the annotion.

dot_size
: A numeric value specifying the size of the dot or a function to calculate the size from the values in the cell or a function to calculate the size from the values in the cell.

dot_size_name
: A character string specifying the name of the legend for the dot size. If NULL, the dot size legend will not be shown.

legend_items
: A numeric vector with names to specifiy the items in the main legend. The names will be working as the labels of the legend items.

legend_discrete
: A logical value indicating whether the main legend is discrete.

legend.position
: Legend position: "none", "left", "right", "bottom", "top"

legend.direction
: Legend direction: "horizontal" or "vertical"

lower_quantile, upper_quantile, lower_cutoff, upper_cutoff
: Vector of minimum and maximum cutoff values or quantile values for each feature. It's applied to aggregated values when aggregated values are used (e.g. plot_type tile, label, etc). It's applied to raw values when raw values are used (e.g. plot_type bars, etc).

add_bg
: A logical value indicating whether to add a background to the heatmap. Does not work with cell_type = "bars" or cell_type = "tile".

bg_alpha
: A numeric value between 0 and 1 specifying the transparency of the background.

add_reticle
: A logical value indicating whether to add a reticle to the heatmap.

`reticle_color` A character string specifying the color of the reticle.

`column_name_annotation`
A logical value indicating whether to add the column annotation for the column names. which is a simple annotaion indicating the column names.

`column_name_legend`
A logical value indicating whether to show the legend of the column name annotation.

`row_name_annotation`
A logical value indicating whether to add the row annotation for the row names. which is a simple annotaion indicating the row names.

`row_name_legend`
A logical value indicating whether to show the legend of the row name annotation.

`cluster_columns`
A logical value indicating whether to cluster the columns. If TRUE and columns_split_by is provided, the clustering will only be applied to the columns within the same split.

`cluster_rows` A logical value indicating whether to cluster the rows. If TRUE and rows_split_by is provided, the clustering will only be applied to the rows within the same split.

`show_row_names` A logical value indicating whether to show the row names. If TRUE, the legend of the row group annotation will be hidden.

`show_column_names`
A logical value indicating whether to show the column names. If TRUE, the legend of the column group annotation will be hidden.

`border` A logical value indicating whether to draw the border of the heatmap. If TRUE, the borders of the slices will be also drawn.

`title` The global (column) title of the heatmap

`column_title` A character string/vector of the column name(s) to use as the title of the column group annotation.

`row_title` A character string/vector of the column name(s) to use as the title of the row group annotation.

`na_col` A character string specifying the color for missing values. The default is "grey85".

`row_names_side` A character string specifying the side of the row names. The default is "right".

`column_names_side`
A character string specifying the side of the column names. The default is "bottom".

`column_annotation`
A character string/vector of the column name(s) to use as the column annotation. Or a list with the keys as the names of the annotation and the values as the column names.

`column_annotation_side`
A character string specifying the side of the column annotation. Could be a list with the keys as the names of the annotation and the values as the sides.

`column_annotation_palette`
A character string specifying the palette of the column annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

column_annotation_palcolor
: A character vector of colors to override the palette of the column annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

column_annotation_type
: A character string specifying the type of the column annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the column data.

column_annotation_params
: A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters passed to the annotation function. For the parameters for names (columns_by, rows_by, columns_split_by, rows_split_by), the key should be "name.(name)", where `(name)` is the name of the annotation. See `anno_pie()`, `anno_ring()`, `anno_bar()`, `anno_violin()`, `anno_boxplot()`, `anno_density()`, `anno_simple()`, `anno_points()` and `anno_lines()` for the parameters of each annotation function.

column_annotation_agg
: A function to aggregate the values in the column annotation.

row_annotation
: A character string/vector of the column name(s) to use as the row annotation. Or a list with the keys as the names of the annotation and the values as the column names.

row_annotation_side
: A character string specifying the side of the row annotation. Could be a list with the keys as the names of the annotation and the values as the sides.

row_annotation_palette
: A character string specifying the palette of the row annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

row_annotation_palcolor
: A character vector of colors to override the palette of the row annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

row_annotation_type
: A character string specifying the type of the row annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the row data.

row_annotation_params
: A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters. Same as `column_annotation_params`.

row_annotation_agg
: A function to aggregate the values in the row annotation.

flip
: A logical value indicating whether to flip the heatmap. The idea is that, you can simply set `flip = TRUE` to flip the heatmap. You don't need to swap the arguments related to rows and columns, except those you specify via `...` that are passed to `ComplexHeatmap::Heatmap()` directly.

alpha	A numeric value between 0 and 1 specifying the transparency of the heatmap cells.
seed	Random seed for reproducibility
layer_fun_callback	A function to add additional layers to the heatmap. The function should have the following arguments: j, i, x, y, w, h, fill, sr and sc. Please also refer to the layer_fun argument in ComplexHeatmap::Heatmap.
cell_type	A character string specifying the type of the heatmap cells. The default is values. Other options are "bars", "label", "dot", "violin", "boxplot". Note that for pie chart, the values under columns specified by rows will not be used directly. Instead, the values will just be counted in different pie_group_by groups. NA values will not be counted.
cell_agg	A function to aggregate the values in the cell, for the cell type "tile" and "label". The default is mean.
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
...	Other arguments passed to ComplexHeatmap::Heatmap() When row_names_max_width is passed, a unit is expected. But you can also pass a numeric values, with a default unit "inches", or a string like "5inches" to specify the number and unit directly.

See Also

anno_simple, anno_points, anno_lines, anno_pie, anno_violin, anno_boxplot, anno_density

Examples

```
set.seed(8525)

matrix_data <- matrix(rnorm(60), nrow = 6, ncol = 10)
rownames(matrix_data) <- paste0("R", 1:6)
colnames(matrix_data) <- paste0("C", 1:10)
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(matrix_data)
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # use a different color palette
  # change the main legend title
  # show row names (legend will be hidden)
  # show column names
  # change the row name annotation name and side
  # change the column name annotation name
  Heatmap(matrix_data,
```

```
    palette = "viridis", values_by = "z-score",
    show_row_names = TRUE, show_column_names = TRUE,
    rows_name = "Features", row_names_side = "left",
    columns_name = "Samples"
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # flip the heatmap
  Heatmap(matrix_data,
    palette = "viridis", values_by = "z-score",
    show_row_names = TRUE, show_column_names = TRUE,
    rows_name = "Features", row_names_side = "left",
    columns_name = "Samples", flip = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # add annotations to the heatmap
  rows_data <- data.frame(
    rows = paste0("R", 1:6),
    group = sample(c("X", "Y", "Z"), 6, replace = TRUE)
  )
  Heatmap(matrix_data,
    rows_data = rows_data,
    row_annotation = list(Group = "group"),
    row_annotation_type = list(Group = "simple"),
    row_annotation_palette = list(Group = "Spectral")
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(matrix_data,
    rows_data = rows_data,
    rows_split_by = "group"
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # add labels to the heatmap
  Heatmap(matrix_data,
    rows_data = rows_data,
    rows_split_by = "group", cell_type = "label",
    label = function(x) {
      ifelse(
        x > 0, scales::number(x, accuracy = 0.01), NA
      )
    }
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # add labels based on an external data
  pvalues <- matrix(runif(60, 0, 0.5), nrow = 6, ncol = 10)
  Heatmap(matrix_data,
    rows_data = rows_data,
    rows_split_by = "group", cell_type = "label",
    label = function(x, i, j) {
      pv <- ComplexHeatmap::pindex(pvalues, i, j)
      ifelse(pv < 0.01, "***",
        ifelse(pv < 0.05, "**",
          ifelse(pv < 0.1, "*", NA)
```

```
        )
      )
    }
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # quickly simulate a GO board
  go <- matrix(sample(c(0, 1, NA), 81, replace = TRUE), ncol = 9)

  Heatmap(
    go,
    # Do not cluster rows and columns and hide the annotations
    cluster_rows = FALSE, cluster_columns = FALSE,
    row_name_annotation = FALSE, column_name_annotation = FALSE,
    show_row_names = FALSE, show_column_names = FALSE,
    # Set the legend items
    values_by = "Players", legend_discrete = TRUE,
    legend_items = c("Player 1" = 0, "Player 2" = 1),
    # Set the pawns
    cell_type = "dot", dot_size = function(x) ifelse(is.na(x), 0, 10),
    dot_size_name = NULL, # hide the dot size legend
    palcolor = c("white", "black"),
    # Set the board
    add_reticle = TRUE,
    # Set the size of the board
    width = ggplot2::unit(105, "mm"), height = ggplot2::unit(105, "mm")
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # Make the row/column name annotation thicker
  Heatmap(matrix_data,
    # Use the "name." prefix
    column_annotation_params = list(name.columns = list(height = 5)),
    row_annotation_params = list(name.rows = list(width = 5))
  )
}

# Use long form data
N <- 500
data <- data.frame(
  value = rnorm(N),
  c = sample(letters[1:8], N, replace = TRUE),
  r = sample(LETTERS[1:5], N, replace = TRUE),
  p = sample(c("x", "y"), N, replace = TRUE),
  q = sample(c("X", "Y", "Z"), N, replace = TRUE),
  a = as.character(sample(1:5, N, replace = TRUE)),
  p1 = runif(N),
  p2 = runif(N)
)

if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
    rows_by = "r", columns_by = "c", values_by = "value",
    rows_split_by = "p", columns_split_by = "q", show_column_names = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
```

```
  # split into multiple heatmaps
  Heatmap(data,
    values_by = "value", columns_by = "c", rows_by = "r", split_by = "p",
    upper_cutoff = 2, lower_cutoff = -2, legend.position = c("none", "right"),
    design = "AAAAAA#BBBBBBB"
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # cell_type = "bars" (default is "tile")
  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "bars"
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  p <- Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "dot", dot_size = length, dot_size_name = "data points",
    add_bg = TRUE, add_reticle = TRUE
  )
  p
}
if (requireNamespace("cluster", quietly = TRUE)) {
  dot_size_data <- p@data
  # Make it big so we can see if we get the right indexing
  # for dot_size function
  dot_size_data["A", "a"] <- max(dot_size_data) * 2

  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "dot", dot_size_name = "data points",
    dot_size = function(x, i, j) ComplexHeatmap::pindex(dot_size_data, i, j),
    show_row_names = TRUE, show_column_names = TRUE,
    add_bg = TRUE, add_reticle = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "pie", pie_group_by = "q", pie_size = sqrt,
    add_bg = TRUE, add_reticle = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "violin", add_bg = TRUE, add_reticle = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    cell_type = "boxplot", add_bg = TRUE, add_reticle = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
```

```
    values_by = "value", rows_by = "r", columns_by = "c",
    column_annotation = list(r1 = "p", r2 = "q", r3 = "p1"),
    column_annotation_type = list(r1 = "ring", r2 = "bar", r3 = "violin"),
    column_annotation_params = list(
      r1 = list(height = grid::unit(10, "mm"), show_legend = FALSE),
      r3 = list(height = grid::unit(18, "mm"))
    ),
    row_annotation = c("q", "p2", "a"),
    row_annotation_side = "right",
    row_annotation_type = list(q = "pie", p2 = "density", a = "simple"),
    row_annotation_params = list(q = list(width = grid::unit(12, "mm"))),
    show_row_names = TRUE, show_column_names = TRUE
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  Heatmap(data,
    values_by = "value", rows_by = "r", columns_by = "c",
    split_by = "p", palette = list(x = "Reds", y = "Blues")
  )
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # implies in_form = "wide-rows"
  Heatmap(data, rows_by = c("p1", "p2"), columns_by = "c")
}
if (requireNamespace("cluster", quietly = TRUE)) {
  # implies wide-columns
  Heatmap(data, rows_by = "r", columns_by = c("p1", "p2"))
}
```

`Histogram` *Histogram*

Description

Creates histograms to illustrate the distribution of continuous data. Supports grouping, faceting, splitting, and trend lines.

Usage

```
Histogram(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  bins = NULL,
  binwidth = NULL,
```

```
  trend_skip_zero = FALSE,
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  position = "identity",
  use_trend = FALSE,
  add_trend = FALSE,
  trend_alpha = 1,
  trend_linewidth = 0.8,
  trend_pt_size = 1.5,
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.5,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  legend.position = waiver(),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for values (numeric expected)
group_by	Column to group the data
group_by_sep	Separator for concatenating multiple group_by columns
group_name	Legend title for group_by
xtrans	Transformation for x-axis (e.g., "identity", "log10")
ytrans	Transformation for y-axis
split_by	Column to split data into multiple plots

split_by_sep	Separator for concatenating multiple split_by columns
flip	Whether to flip the plot
bins	Number of bins for histogram
binwidth	Width of bins for histogram
trend_skip_zero	Skip zero counts in trend line
add_bars	Whether to add data distribution lines at bottom
bar_height	Height of distribution bars (as fraction of max)
bar_alpha	Alpha for distribution bars
bar_width	Width of distribution bars
position	Position adjustment for overlapping groups
use_trend	Use trend line instead of histogram bars
add_trend	Add trend line to histogram
trend_alpha	Alpha for trend line and points
trend_linewidth	Width of trend line
trend_pt_size	Size of trend points
palette	Palette name
palcolor	Custom colors
alpha	Transparency for density curves
theme	Theme name or function
theme_args	Arguments passed to theme function
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
expand	Axis expansion
facet_by	Columns for faceting
facet_scales	Facet scales type
facet_ncol	Number of facet columns
facet_nrow	Number of facet rows
facet_byrow	Fill facets by row
legend.position	Legend position
legend.direction	Legend direction
seed	Random seed
combine	Whether to combine multiple plots
nrow	Number of rows for combined plots
ncol	Number of columns for combined plots
byrow	Fill combined plots by row
axes	Axis handling for combined plots
axis_titles	Axis title handling for combined plots
guides	Guide handling for combined plots
design	Custom design for combined plots
...	Additional arguments passed to geom_density

Value

A ggplot object, combined plot, or list of plots

Examples

```
set.seed(8525)
data <- data.frame(
  x = sample(setdiff(1:100, c(30:36, 50:55, 70:77)), 1000, replace = TRUE),
  group = factor(rep(c("A", "B"), each = 500), levels = c("A", "B")),
  facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

Histogram(data, x = "x")
Histogram(data, x = "x", group_by = "group")
Histogram(data, x = "x", split_by = "facet", add_bars = TRUE)
Histogram(data, x = "x", group_by = "group", add_trend = TRUE)
```

JitterPlot *Jitter Plot*

Description

Jittered point plot with optional background, highlight, labels and faceting. Supports point sizing, dodge grouping, and various customization options.

Usage

```
JitterPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
 sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  keep_empty = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_text_angle = 0,
  order_by = "-({y}^2 + {size_by}^2)",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  shape = 21,
  border = "black",
  size_by = 2,
  size_name = NULL,
  size_trans = NULL,
  y_nbreaks = 4,
  jitter_width = 0.5,
  jitter_height = 0,
  y_max = NULL,
  y_min = NULL,
  y_trans = "identity",
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_hline = NULL,
  hline_type = "solid",
  hline_width = 0.5,
  hline_color = "black",
  hline_alpha = 1,
  labels = NULL,
  label_by = NULL,
  nlabel = 5,
  label_size = 3,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	A character string of the column name to plot on the x-axis. A character/factor column is expected. If multiple columns are provided, the columns will be concatenated with x_sep.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided. When in_form is "wide", x columns will not be concatenated.
y	A character string of the column name to plot on the y-axis. A numeric column is expected. When in_form is "wide", y is not required. The values under x columns will be used as y-values.
in_form	A character string to specify the input data type. Either "long" or "wide".
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
sort_x	A character string to specify the sorting of x-axis, chosen from "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median". • none means no sorting (as-is). • mean_asc sorts the x-axis by ascending mean of y-values. • mean_desc sorts the x-axis by descending mean of y-values. • mean is an alias for mean_asc. • median_asc sorts the x-axis by ascending median of y-values. • median_desc sorts the x-axis by descending median of y-values. • median is an alias for median_asc.
flip	A logical value to flip the plot.
keep_empty	A logical value to keep the empty levels in the x-axis.
group_by	A character string to dodge the points.
group_by_sep	A character string to concatenate the columns in group_by, if multiple columns are provided.
group_name	A character string to name the legend of dodge.
x_text_angle	Angle for x-axis text labels
order_by	A string of expression passed to dplyr::arrange() to order the data to get the top nlabel points for labeling. Default is -({y}^2 + {size_by}^2) (similar to VolcanoPlot).
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
shape	A numeric value to specify the point shape. Shapes 21–25 have borders; border behavior is controlled by border.

border	A logical or character value to specify the border of points when the shape has border (21–25). If TRUE, border color follows the point color (same as fill). If a color string, uses that constant border color. If FALSE, no border.
size_by	A numeric column name or a single numeric value for the point size. When a column, sizes are scaled (see scatter plots).
size_name	Legend title for size when size_by is a column.
size_trans	A function or a name of a global function to transform size_by (when size_by is a column). The legend shows original (untransformed) values.
y_max, y_min	Numeric or quantile strings ("q95", "q5") for y limits computation (used for fixed coord).
y_trans, y_nbreaks	Axis settings.
add_bg	A logical value to add background to the plot.
bg_palette	A character string to specify the palette of the background.
bg_palcolor	A character vector to specify the colors of the background.
bg_alpha	A numeric value to specify the transparency of the background.
add_hline	Add one or more horizontal reference lines at the given y-value(s).
hline_type	The line type for the horizontal reference line(s).
hline_width	The line width for the horizontal reference line(s).
hline_color	The color for the horizontal reference line(s).
hline_alpha	The alpha for the horizontal reference line(s).
labels	A vector of row names or indices to label the points.
label_by	A character column name to use as the label text. If NULL, rownames are used.
nlabel	Number of points to label per x-group when labels is NULL (top by y^2 + size^2).
label_size, label_fg, label_bg, label_bg_r	Label aesthetics.
highlight, highlight_color, highlight_size, highlight_alpha	Highlighted point options.
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row

axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

The Jitter plot(s). When split_by is not provided, it returns a ggplot object. When split_by is provided, it returns a object of plots wrapped by patchwork::wrap_plots if combine = TRUE; otherwise, it returns a list of ggplot objects.

Examples

```
set.seed(8525)
n <- 200
x <- sample(LETTERS[1:5], n, replace = TRUE)
group <- sample(c("G1", "G2"), n, replace = TRUE)
size <- rexp(n, rate = 1)
id <- paste0("pt", seq_len(n))
y <- rnorm(n, mean = ifelse(group == "G1", 0.5, -0.5)) +
  as.numeric(factor(x, levels = LETTERS[1:5])) / 10
df <- data.frame(
  x = factor(x, levels = LETTERS[1:5]),
  y = y,
  group = group,
  size = size,
  id = id
)

# Basic
JitterPlot(df, x = "x", y = "y")

# Map size with transform; legend shows original values
JitterPlot(df,
  x = "x", y = "y", size_by = "size", size_name = "Abundance",
  size_trans = sqrt, order_by = "-y^2"
)

# Dodge by group and add a horizontal line
JitterPlot(df,
  x = "x", y = "y", group_by = "group",
  add_hline = 0, hline_type = "dashed", hline_color = "red2"
)

# Label top points by distance (y^2 + size^2)
JitterPlot(df, x = "x", y = "y", size_by = "size", label_by = "id", nlabel = 3)

# Flip axes
JitterPlot(df, x = "x", y = "y", flip = TRUE)
```

KMPlot *Kaplan-Meier Survival Plot*

Description

Create publication-ready Kaplan-Meier survival curves with optional risk tables, confidence intervals, and statistical comparisons.

This function provides a complete implementation of Kaplan-Meier survival analysis visualization, supporting single or multiple groups, with automatic p-value calculation using the log-rank test.

Usage

```
KMPlot(
  data,
  time,
  status,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  show_pval = TRUE,
  pval_method = "logrank",
  pval_digits = 4,
  pval_size = 4.5,
  pval_coord = c(0.05, 0.1),
  show_conf_int = FALSE,
  conf_alpha = 0.2,
  show_median_line = "none",
  median_linetype = 2,
  median_linewidth = 0.6,
  line_width = 1.3,
  show_risk_table = FALSE,
  risk_table_height = 0.25,
  risk_table_fontsize = 3.5,
  show_censors = TRUE,
  censor_shape = 3,
  censor_size = 4,
  censor_stroke = 0.5,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
```

```
  x_breaks = NULL,
  y_breaks = waiver(),
  x_min = NULL,
  x_max = NULL,
  y_min = 0,
  y_max = 1,
  legend.position = "top",
  legend.direction = "horizontal",
  title = NULL,
  subtitle = NULL,
  xlab = "Time",
  ylab = "Survival Probability",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
time	Column name for time variable (numeric).
status	Column name for event status (1=event, 0=censored).
group_by	Column(s) for grouping survival curves.
group_by_sep	Separator for concatenating multiple group columns.
group_name	Legend title for groups.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
show_pval	Show log-rank test p-value.
pval_method	P-value calculation method ("logrank").
pval_digits	Number of digits for p-value.
pval_size	Text size for p-value.
pval_coord	Position of p-value as c(x, y) where x is fraction of x-axis, y is absolute.
show_conf_int	Show confidence interval ribbons.
conf_alpha	Transparency for confidence interval ribbons.

show_median_line
: Show median survival lines: "none", "h", "v", "hv".

median_linetype
: Line type for median survival lines.

median_linewidth
: Line width for median survival lines.

line_width
: Width of survival curves.

show_risk_table
: Show risk table below plot.

risk_table_height
: Relative height of risk table (0-1).

risk_table_fontsize
: Font size for numbers in risk table.

show_censors
: Show censoring marks on curves.

censor_shape
: Shape for censoring marks.

censor_size
: Size for censoring marks.

censor_stroke
: Stroke width for censoring marks.

theme
: Theme name (string) or theme function

theme_args
: List of arguments passed to theme function

palette
: Color palette name

palcolor
: Custom colors for palette

alpha
: Transparency level (0-1)

aspect.ratio
: Aspect ratio of plot panel

x_breaks
: Custom x-axis breaks (time points).

y_breaks
: Custom y-axis breaks.

x_min
: Minimum x-axis value.

x_max
: Maximum x-axis value.

y_min
: Minimum y-axis value (default: 0).

y_max
: Maximum y-axis value (default: 1).

legend.position
: Legend position: "none", "left", "right", "bottom", "top"

legend.direction
: Legend direction: "horizontal" or "vertical"

title
: Plot title

subtitle
: Plot subtitle

xlab
: X-axis label

ylab
: Y-axis label

combine
: Whether to combine split plots into one

nrow
: Number of rows when combining plots

ncol
: Number of columns when combining plots

byrow
: Fill combined plots by row

seed
: Random seed for reproducibility

axes
: How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")

axis_titles
: How to handle axis titles in combined plots

guides
: How to handle guides in combined plots ("collect", "keep", "auto")

design
: Custom layout design for combined plots

Value

A ggplot object, list of plots, or combined plots

Examples

```
library(survival)

# Basic Kaplan-Meier plot
KMPlot(data = lung, time = "time", status = "status")

# Multiple groups with p-value
KMPlot(
  data = lung,
  time = "time",
  status = "status",
  group_by = "sex",
  show_pval = TRUE
)

# With risk table
KMPlot(
  data = lung,
  time = "time",
  status = "status",
  group_by = "sex",
  show_risk_table = TRUE,
  show_pval = TRUE
)

# With confidence intervals and median lines
KMPlot(
  data = lung,
  time = "time",
  status = "status",
  group_by = "sex",
  show_conf_int = TRUE,
  show_median_line = "hv",
  palette = "Set1"
)

# Publication-ready plot
KMPlot(
  data = lung,
  time = "time",
  status = "status",
  group_by = "sex",
  show_risk_table = TRUE,
  show_pval = TRUE,
  show_conf_int = TRUE,
  show_median_line = "hv",
  palette = "jco",
  title = "Overall Survival by Sex",
  xlab = "Time (days)",
  ylab = "Survival Probability",
  theme_args = list(base_size = 14)
)
```

LinePlot *Line Plot*

Description

Visualizing the change of a numeric value over the progression of a categorical variable. Supports both single lines and grouped lines with extensive customization options.

Usage

```
LinePlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  fill_point_by_x_if_no_group = TRUE,
  color_line_by_x_if_no_group = TRUE,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_errorbars = FALSE,
  errorbar_width = 0.1,
  errorbar_alpha = 1,
  errorbar_color = "grey30",
  errorbar_linewidth = 0.75,
  errorbar_min = NULL,
  errorbar_max = NULL,
  errorbar_sd = NULL,
  highlight = NULL,
  highlight_size = NULL,
  highlight_color = "red2",
  highlight_alpha = 0.8,
  pt_alpha = 1,
  pt_size = 5,
  line_type = "solid",
  line_width = 1,
  line_alpha = 0.8,
  add_hline = FALSE,
  hline_type = "solid",
  hline_width = 0.5,
  hline_color = "black",
  hline_alpha = 1,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
```

```
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column for x-axis (categorical)
y	Column for y-axis (numeric). If NULL, counts will be used
x_sep	Separator for concatenating multiple x columns
group_by	Column to group lines by. If NULL, creates a single line
group_by_sep	Separator for concatenating multiple group_by columns
group_name	Legend name for groups
fill_point_by_x_if_no_group	Logical. If TRUE and no group_by, color points by x values
color_line_by_x_if_no_group	Logical. If TRUE and no group_by, color lines by x values
add_bg	Logical. Add background stripes
bg_palette	Palette for background
bg_palcolor	Custom colors for background
bg_alpha	Alpha value for background
add_errorbars	Logical. Add error bars
errorbar_width	Width of error bars

errorbar_alpha	Alpha value for error bars
errorbar_color	Color for error bars. Use "line" to match line colors
errorbar_linewidth	Line width for error bars
errorbar_min	Column containing lower error bounds
errorbar_max	Column containing upper error bounds
errorbar_sd	Column containing standard deviation. Used if min/max not provided
highlight	Rows to highlight. Can be numeric index, rownames, or expression string
highlight_size	Size of highlighted points
highlight_color	Color for highlighted points
highlight_alpha	Alpha for highlighted points
pt_alpha	Alpha value for points
pt_size	Size of points
line_type	Line type
line_width	Line width
line_alpha	Alpha value for lines
add_hline	Y-intercept for horizontal line(s). Can be numeric or named list for groups
hline_type	Line type for horizontal line
hline_width	Line width for horizontal line
hline_color	Color for horizontal line. Use TRUE to match group colors
hline_alpha	Alpha for horizontal line
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_nrow	Number of rows in facet layout
facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
x_text_angle	Angle for x-axis text
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title

subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Keep empty factor levels
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or combined plots

Examples

```
# Basic line plot
data <- data.frame(
  time = rep(c("T1", "T2", "T3", "T4"), 2),
  value = c(10, 15, 13, 20, 8, 12, 11, 18),
  group = rep(c("Control", "Treatment"), each = 4)
)
LinePlot(data, x = "time", y = "value")

# Grouped line plot
LinePlot(data, x = "time", y = "value", group_by = "group")

# With background stripes
LinePlot(data, x = "time", y = "value", group_by = "group", add_bg = TRUE)

# With error bars
data$sd <- c(2, 2.5, 2, 3, 1.5, 2, 1.8, 2.5)
LinePlot(
  data,
  x = "time",
  y = "value",
  group_by = "group",
  add_errorbars = TRUE,
  errorbar_sd = "sd"
)

# With horizontal reference line
LinePlot(
  data,
  x = "time",
  y = "value",
  group_by = "group",
```

```
  add_hline = 15,
  hline_type = "dashed"
)

# Split by another variable
data$batch <- rep(c("Batch1", "Batch2"), each = 4)
LinePlot(data, x = "time", y = "value", group_by = "group", split_by = "batch")

# Faceted plot
LinePlot(data, x = "time", y = "value", group_by = "group", facet_by = "batch")
```

`LollipopPlot` *Lollipop Plot*

Description

Creates a lollipop plot with a numeric x-axis and categorical y-axis. This is a convenience wrapper around `DotPlot` with `lollipop = TRUE`.

Usage

```
LollipopPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_reverse = FALSE,
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  theme = "theme_ggforge_grid",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
```

```
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	A character string specifying the column to use for the x-axis (numeric column expected).
y	A character string specifying the column to use for the y-axis (factor/character column expected).
y_sep	A character string to concatenate multiple columns in y.
flip	Whether to flip the x and y axes.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
size_by	Which column to use as the size of the dots (numeric column). If not provided, the size will be the count of instances for each (x, y) pair. Can also be a single numeric value to specify a fixed size.
fill_by	Which column to use to fill the dots (numeric column). If not provided, all dots will be filled with the middle color of the palette.
fill_cutoff	A numeric value specifying the cutoff for the fill column. Values below (or above if `fill_reverse = TRUE`) will be shown in grey.
fill_reverse	Whether to reverse the fill direction. If FALSE (default), values < cutoff are grey. If TRUE, values > cutoff are grey.
size_name	A character string to name the size legend.
fill_name	A character string to name the fill legend.
fill_cutoff_name	A character string to name the fill cutoff legend.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"

facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
x_text_angle	Angle for x-axis text.
seed	Random seed for reproducibility
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Whether to keep empty factor levels.
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
mtcars <- datasets::mtcars
LollipopPlot(mtcars,
  x = "qsec", y = "gear", size_by = "wt",
  fill_by = "mpg"
)
LollipopPlot(mtcars,
  x = "qsec", y = "gear", size_by = "wt",
  fill_by = "mpg", fill_cutoff = 18
)
```

ManhattanPlot *Manhattan Plot*

Description

Creates a Manhattan plot for visualizing genome-wide association study (GWAS) results. This function is adapted from ggmanh::manhattan_plot() with several enhancements:

- Parameter names use underscores instead of dots for consistency
- chr.colname, pos.colname, pval.colname and label.colname are renamed to chr_by, pos_by, pval_by and label_by
- The chromosome and chr.order arguments are merged into chromosomes
- The highlight.colname argument is replaced with highlight, which can be a vector of indices or a character expression to select variants, instead of a column name
- Point styling controlled by pt_* arguments (size, color, alpha, shape)
- Label styling controlled by label_* arguments
- Highlight styling controlled by highlight_* arguments
- Flexible p-value transformation via pval_transform function
- Improved significance threshold handling

Usage

```
ManhattanPlot(
  data,
  chr_by,
  pos_by,
  pval_by,
  split_by = NULL,
  split_by_sep = "_",
  label_by = NULL,
  chromosomes = NULL,
  pt_size = 0.75,
  pt_color = NULL,
  pt_alpha = alpha,
  pt_shape = 19,
  label_size = 3,
  label_fg = NULL,
  highlight = NULL,
  highlight_color = NULL,
  highlight_size = 1.5,
  highlight_alpha = 1,
  highlight_shape = 19,
  preserve_position = TRUE,
  chr_gap_scaling = 1,
  pval_transform = "-log10",
  signif = c(5e-08, 1e-05),
  signif_color = NULL,
  signif_rel_pos = 0.2,
  signif_label = TRUE,
```

```
  signif_label_size = 3.5,
  signif_label_pos = c("left", "right"),
  thin = NULL,
  thin_n = 1000,
  thin_bins = 200,
  rescale = TRUE,
  rescale_ratio_threshold = 5,
  palette = "Dark2",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = expression("-" * log[10](p)),
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  facet_by = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame or GenomicRanges::GRanges containing the data to be plotted.
chr_by	Column name for chromosome (default: "chr").
pos_by	Column name for position (default: "pos").
pval_by	Column name for p-value (default: "pval").
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
label_by	Column name for the variants to be labeled (default: NULL). Only the variants with non-empty values in this column will be labeled.
chromosomes	A vector of chromosomes to be plotted (default: NULL). If NULL, all chromosomes will be plotted. Can be used to select chromosomes to be plotted or to set the order of the chromosomes.
pt_size	A numeric value to specify the size of the points in the plot.
pt_color	A character string to specify the color of the points in the plot. By default, the color of the points will be controlled by palette or palcolor arguments. This is useful to color the background points when highlight and highlight_color are specified.
pt_alpha	A numeric value to specify the transparency of the points in the plot.
pt_shape	A numeric value to specify the shape of the points in the plot.

label_size A numeric value to specify the size of the labels in the plot.

label_fg A character string to specify the color of the labels in the plot. If NULL, the color of the labels will be the same as the points.

highlight Either a vector of indices or a character expression to select the variants to be highlighted (default: NULL). If NULL, no variants will be highlighted.

highlight_color
A character string to specify the color of the highlighted points.

highlight_size A numeric value to specify the size of the highlighted points.

highlight_alpha
A numeric value to specify the transparency of the highlighted points.

highlight_shape
A numeric value to specify the shape of the highlighted points.

preserve_position
If TRUE, the width of each chromosome reflects the number of variants and the position of each variant is correctly scaled. If FALSE, the width of each chromosome is equal and the variants are equally spaced.

chr_gap_scaling
A numeric value to specify the scaling of the gap between chromosomes. It is used to adjust the gap between chromosomes in the plot.

pval_transform A function to transform the p-values (default: "-log10"). If it is a character, it will be evaluated as a function.

signif A vector of significance thresholds (default: c(5e-08, 1e-05)).

signif_color A character vector of equal length as signif. It contains colors for the lines drawn at signif. If NULL, the smallest value is colored black while others are grey.

signif_rel_pos A numeric between 0.1 and 0.9. If the plot is rescaled, where should the significance threshold be positioned?

signif_label A logical value indicating whether to label the significance thresholds (default: TRUE).

signif_label_size
A numeric value to specify the size of the significance labels.

signif_label_pos
A character string specifying the position of the significance labels. It can be either "left" or "right" (default: "left").

thin A logical value indicating whether to thin the data (default: NULL). Defaults to TRUE when chromosomes is specified and the length of it is less than the number of chromosomes in the data. Defaults to FALSE otherwise.

thin_n Number of max points per horizontal partitions of the plot. Defaults to 1000.

thin_bins Number of bins to partition the data. Defaults to 200.

rescale A logical value indicating whether to rescale the plot (default: TRUE).

rescale_ratio_threshold
A numeric value to specify the ratio threshold for rescaling.

palette Color palette name

palcolor Custom colors for palette

palreverse A logical value indicating whether to reverse the palette for chromosomes (default: FALSE).

alpha Transparency level (0-1)

theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
facet_by	Column name(s) for faceting the plot
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects. If no `split_by` is provided, a single plot (ggplot object) will be returned. If 'combine' is TRUE, a wrap_plots object will be returned. If 'combine' is FALSE, a list of ggplot objects will be returned.

Examples

```
set.seed(1000)

nsim <- 50000

simdata <- data.frame(
  "chromosome" = sample(c(1:22, "X"), size = nsim, replace = TRUE),
  "position" = sample(1:100000000, size = nsim),
  "P.value" = rbeta(nsim, shape1 = 5, shape2 = 1)^7,
  "cohort" = sample(c("A", "B"), size = nsim, replace = TRUE)
)
simdata$chromosome <- factor(simdata$chromosome, c(1:22, "X"))

if (requireNamespace("ggmanh", quietly = TRUE)) {
  ManhattanPlot(
    simdata,
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated P.Values", ylab = "P"
  )
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
  # split_by
  ManhattanPlot(
    simdata,
```

```
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated P.Values", ylab = "P", split_by = "cohort", ncol = 1
  )
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
  # Customized p-value transformation and significance threshold line colors
  ManhattanPlot(
    simdata,
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated -Log2 P.Values", ylab = "-log2(P)", pval_transform = "-log2",
    signif_color = c("red", "blue")
  )
}
```

`Network` *Network Plot*

Description

Visualize network graphs with nodes and links using igraph layouts and ggraph. Supports directed and undirected graphs, customizable node and link aesthetics, clustering, and various layout algorithms.

Usage

```
Network(
  links,
  nodes = NULL,
  split_by = NULL,
  split_by_sep = "_",
  split_nodes = FALSE,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  node_by = NULL,
  node_by_sep = "_",
  link_weight_by = 2,
  link_weight_name = NULL,
  link_type_by = "solid",
  link_type_name = NULL,
  node_size_by = 15,
  node_size_name = NULL,
  node_color_by = "black",
  node_color_name = NULL,
  node_shape_by = 21,
  node_shape_name = NULL,
  node_fill_by = "grey20",
  node_fill_name = NULL,
  link_alpha = 1,
```

```
  node_alpha = 0.95,
  node_stroke = 1.5,
  cluster_scale = c("fill", "color", "shape"),
  node_size_range = c(5, 20),
  link_weight_range = c(0.5, 5),
  link_arrow_offset = 20,
  link_curvature = 0,
  link_color_by = "from",
  link_color_name = NULL,
  palette = "Paired",
  palcolor = NULL,
  link_palette = ifelse(link_color_by %in% c("from", "to"), palette, "Set1"),
  link_palcolor = if (link_color_by %in% c("from", "to")) palcolor else NULL,
  directed = TRUE,
  layout = "circle",
  cluster = "none",
  add_mark = FALSE,
  mark_expand = ggplot2::unit(10, "mm"),
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  add_label = TRUE,
  label_size = 3,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  arrow = ggplot2::arrow(type = "closed", length = ggplot2::unit(0.1, "inches")),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

links	A data frame containing the links between nodes.
nodes	A data frame containing the nodes (optional). Node names are extracted from

	the links data frame if not provided. If "@nodes" is provided, the nodes data frame will be extracted from the nodes attribute of the links data frame.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
split_nodes	Whether to split nodes data when splitting by a column.
from	Column name(s) for source nodes. Default is the first column.
from_sep	Separator for concatenating multiple from columns.
to	Column name(s) for target nodes. Default is the second column.
to_sep	Separator for concatenating multiple to columns.
node_by	Column name(s) for node identifiers. Default is the first column.
node_by_sep	Separator for concatenating multiple node_by columns.
link_weight_by	Numeric value or column name for link width.
link_weight_name	Legend title for link weight.
link_type_by	Link line type: "solid", "dashed", "dotted", or a column name.
link_type_name	Legend title for link type.
node_size_by	Numeric value or column name for node size.
node_size_name	Legend title for node size.
node_color_by	Color value or column name for node color (border).
node_color_name	Legend title for node color.
node_shape_by	Numeric value or column name for node shape.
node_shape_name	Legend title for node shape.
node_fill_by	Fill color value or column name for node fill.
node_fill_name	Legend title for node fill.
link_alpha	Transparency for links (0-1).
node_alpha	Transparency for node fill (0-1).
node_stroke	Width of node borders.
cluster_scale	Which aesthetic to use for clusters: "fill", "color", or "shape".
node_size_range	Range for node sizes.
link_weight_range	Range for link widths.
link_arrow_offset	Offset for link arrows to avoid overlapping nodes.
link_curvature	Curvature of links (0 = straight).
link_color_by	Link coloring: "from", "to", or a column name.
link_color_name	Legend title for link color.
palette	Color palette name
palcolor	Custom colors for palette
link_palette	Palette for link colors.

link_palcolor	Custom colors for link palette.
directed	Whether the graph is directed.
layout	Layout algorithm: "circle", "tree", "grid", or an igraph layout name.
cluster	Clustering method: "none", "fast_greedy", "walktrap", "edge_betweenness", "infomap", or an igraph clustering function.
add_mark	Whether to add visual marks around clusters.
mark_expand	Expansion of cluster marks.
mark_type	Type of cluster marks: "hull", "ellipse", "rect", or "circle".
mark_alpha	Transparency of cluster marks.
mark_linetype	Line type of cluster marks.
add_label	Whether to add labels to nodes.
label_size	Size of node labels.
label_fg	Foreground color of labels.
label_bg	Background color of labels.
label_bg_r	Background ratio for labels.
arrow	Arrow specification for directed graphs.
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
aspect.ratio	Aspect ratio of plot panel
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, wrap_plots object, or a list of ggplot objects

Examples

```
## Not run:
actors <- data.frame(
  name = c("Alice", "Bob", "Cecil", "David", "Esmeralda"),
  age = c(48, 33, 45, 34, 21),
  shape = c(21, 22, 21, 22, 23),
  gender = c("F", "M", "F", "M", "F")
)
relations <- data.frame(
  from = c(
    "Bob", "Cecil", "Cecil", "David", "David", "Esmeralda", "Bob", "Alice",
    "Cecil", "David"
  ),
  to = c(
    "Alice", "Bob", "Alice", "Alice", "Bob", "Alice", "Bob", "Alice", "Cecil",
    "David"
  ),
  friendship = c(4, 5, 5, 2, 1, 1, 2, 1, 3, 4),
  type = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)
)
Network(relations, actors)
Network(relations, actors,
  link_weight_by = "friendship", node_size_by = "age",
  node_fill_by = "gender", link_color_by = "to",
  layout = "circle", link_curvature = 0.2
)
Network(relations, actors,
  layout = "tree", directed = FALSE, cluster = "fast_greedy",
  add_mark = TRUE
)

## End(Not run)
```

`palettes` *Color Palette System for ggforge*

Description

A comprehensive color palette system with support for both discrete and continuous palettes, custom colors, and intelligent color mapping.

`palette_list` *Color Palette Collection*

Description

A comprehensive collection of color palettes for data visualization. Includes palettes from RColorBrewer, ggsci, viridis, and custom palettes.

Usage

```
palette_list
```

Format

A named list of color vectors. Each palette has an attribute "type" indicating whether it is "discrete" or "continuous".

See Also

get_palette, show_palettes

PieChart *Pie Chart*

Description

Creates pie charts to illustrate numerical proportion of each group. Supports splitting by groups, faceting, and custom color palettes.

Usage

```
PieChart(
  data,
  x,
  y = NULL,
  label = y,
  split_by = NULL,
  split_by_sep = "_",
  clockwise = TRUE,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
```

```
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame
x	Column name for categories. Will be converted to factor.
y	Column name for values. If NULL, counts will be used.
label	Column to use for labels. Default is same as y. If y is NULL, use ".y" to specify counts as labels.
split_by	Column(s) to split the data by, creating separate plots. Multiple columns will be concatenated with split_by_sep.
split_by_sep	Separator for concatenating multiple split_by columns.
clockwise	Whether to draw pie chart clockwise (default: TRUE)
facet_by	Column(s) to facet by (max 2 columns).
facet_scales	Scale type for facets: "fixed", "free", "free_x", or "free_y"
facet_ncol	Number of columns for faceting.
facet_nrow	Number of rows for faceting.
facet_byrow	Whether to fill facets by row.
theme	Theme name or function. Default is "theme_ggforge".
theme_args	List of arguments to pass to the theme function.
palette	Color palette name. Default is "Paired".
palcolor	Custom colors (overrides palette).
alpha	Transparency level (0-1). Default is 1.
aspect.ratio	Aspect ratio of the plot. Default is 1.
legend.position	Legend position ("none", "left", "right", "bottom", "top").
legend.direction	Legend direction ("horizontal" or "vertical").
title	Plot title. Can be a string or function.
subtitle	Plot subtitle.
xlab	X-axis label.
ylab	Y-axis label.
keep_empty	Keep empty factor levels.
combine	Whether to combine plots when split_by is used.
nrow	Number of rows when combining plots.
ncol	Number of columns when combining plots.
byrow	Whether to arrange plots by row when combining.
seed	Random seed for reproducibility. Default is 8525.
axes	Axis handling when combining plots.
axis_titles	Axis title handling when combining plots.
guides	Guide handling when combining plots.
design	Custom design for combining plots.
...	Additional arguments (currently unused).

Value

A ggplot object, patchwork object (if combine=TRUE), or list of plots

Examples

```
# Create sample data
data <- data.frame(
  x = c("A", "B", "C", "D", "E", "F", "G", "H"),
  y = c(10, 8, 16, 4, 6, 12, 14, 2),
  group = c("G1", "G1", "G2", "G2", "G3", "G3", "G4", "G4"),
  facet = c("F1", "F2", "F3", "F4", "F1", "F2", "F3", "F4")
)

# Basic pie chart
PieChart(data, x = "x", y = "y")

# Counter-clockwise
PieChart(data, x = "x", y = "y", clockwise = FALSE)

# With labels
PieChart(data, x = "x", y = "y", label = "group")

# With faceting
PieChart(data, x = "x", y = "y", facet_by = "facet")

# Split by group
PieChart(data, x = "x", y = "y", split_by = "group")

# Custom palettes per split
PieChart(data,
  x = "x", y = "y", split_by = "group",
  palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purples")
)

# Use counts (y from count)
PieChart(data, x = "group")

# With count labels
PieChart(data, x = "group", label = ".y")
```

QQPlot *QQ Plot*

Description

QQ plot is a graphical tool to compare two distributions by plotting their quantiles against each other. Can also create PP (probability-probability) plots.

Usage

```
QQPlot(
  data,
  val,
```

```
  val_trans = NULL,
  type = c("qq", "pp"),
  band = NULL,
  line = list(),
  point = list(),
  fill_name = "Bands",
  band_alpha = 0.5,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
val	A character string of the column name for the values to plot. A numeric column is expected.
val_trans	A function to transform the values before plotting. Default is NULL, which means no transformation.
type	A character string to specify the type of plot. Default is "qq", which means QQ plot. Other option is "pp" for PP plot.
band	A list of arguments to pass to `qqplotr::stat_qq_band()` or `qqplotr::stat_pp_band()`, depending on the value of `type`. Default is NULL, which means no band. If an empty list or TRUE is provided, the default arguments will be used. Multiple bands can be added by providing a list of lists.

line	A list of arguments to pass to qqplotr::stat_qq_line() or qqplotr::stat_pp_line(), depending on the value of type. Default is list(), which means to add a line with default arguments. If NULL is provided, no line will be added.
point	A list of arguments to pass to qqplotr::stat_qq_point() or qqplotr::stat_pp_point(), depending on the value of type. Default is list(), which means to add points with default arguments. If NULL is provided, no points will be added (not recommended).
fill_name	A character string to name the legend of fill. Default is "Bands".
band_alpha	A numeric value to set the alpha of all bands. Default is 0.5. It is a shortcut for setting alpha of all bands. You can override it by setting alpha in band argument.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_nrow	Number of rows in facet layout
facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlim	A numeric vector of length 2 to set the x-axis limits.
ylim	A numeric vector of length 2 to set the y-axis limits.
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(norm = rnorm(100))

QQPlot(data, val = "norm", band = TRUE)
QQPlot(data, val = "norm", band = list(
  list(bandType = "ks", mapping = ggplot2::aes(fill = "KS"), alpha = 0.3),
  list(bandType = "ts", mapping = ggplot2::aes(fill = "TS")),
  list(bandType = "pointwise", mapping = ggplot2::aes(fill = "Normal")),
  list(bandType = "boot", mapping = ggplot2::aes(fill = "Bootstrap"))
), band_alpha = 0.6)

## Not run:
data(airquality, package = "datasets")
di <- "exp" # exponential distribution
dp <- list(rate = 2) # exponential rate parameter
QQPlot(airquality,
  val = "Ozone",
  band = list(distribution = di, dparams = dp),
  line = list(distribution = di, dparams = dp),
  point = list(distribution = di, dparams = dp)
)

de <- TRUE # enabling the detrend option
QQPlot(airquality,
  val = "Ozone",
  band = list(distribution = di, dparams = dp, detrend = de),
  line = list(distribution = di, dparams = dp, detrend = de),
  point = list(distribution = di, dparams = dp, detrend = de)
)

QQPlot(data, val = "norm", type = "pp", band = TRUE)

dp <- list(mean = 2, sd = 2) # shifted and rescaled Normal parameters
QQPlot(data,
  val = "norm", type = "pp",
  band = list(dparams = dp),
  point = list(dparams = dp)
)

QQPlot(data,
  val = "norm", type = "pp", band = TRUE,
  line = list(ab = c(.2, .5))
)

di <- "exp"
dp <- list(rate = .022) # value is based on some empirical tests
de <- TRUE
QQPlot(airquality,
  val = "Ozone", type = "pp",
  band = list(distribution = di, detrend = de, dparams = dp),
  line = list(detrend = de),
```

```
  point = list(distribution = di, detrend = de, dparams = dp),
  ylim = c(-.5, .5)
)

## End(Not run)
```

RadarPlot *Radar Plot / Spider Plot*

Description

Create a radar plot (circular grid) or spider plot (polygonal grid) for visualizing multivariate data across multiple categories.

Usage

```
RadarPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  bg_color = "grey80",
  bg_alpha = 0.1,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
```

```
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)

SpiderPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  bg_color = "grey80",
  bg_alpha = 0.1,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
```

```
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for the x-axis/circles. Character/factor column expected.
x_sep	Separator for concatenating multiple x columns.
group_by	Column name(s) for grouping data (the lines). Character/factor column(s) expected.
group_by_sep	Separator for concatenating multiple group_by columns.
y	Column name for the y-axis. Numeric column expected. If NULL, counts of x-axis column in each group will be used.
group_name	Legend title for groups.
scale_y	How to scale the y-axis. Options: "group", "global", "x", "none". • "group": Scale to fraction within each group • "global": Scale to fraction of total • "x": Scale to fraction within each x-axis group • "none": Use raw counts/values
y_min	Minimum value of y-axis.
y_max	Maximum value of y-axis.
y_nbreaks	Number of breaks in y-axis.
bg_color	Background color of the plot.
bg_alpha	Transparency of background color.
fill	Fill polygons with colors.
linewidth	Width of the lines.
pt_size	Size of the points.
max_charwidth	Maximum character width for x labels.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_nrow	Number of rows in facet layout
facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette

alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
polygon	Draw polygons instead of circles as panel grid.

Value

A ggplot object, wrapped plots, or list of plots

Examples

```
# Use counts
data <- data.frame(
  x = c(rep("A", 2), rep("B", 3), rep("C", 3), rep("D", 4), rep("E", 5)),
  group = sample(paste0("G", 1:4), 17, replace = TRUE)
)
RadarPlot(data, x = "x")
RadarPlot(data, x = "x", bg_color = "lightpink")
RadarPlot(data, x = "x", scale_y = "none")
RadarPlot(data, x = "x", group_by = "group")
SpiderPlot(data, x = "x")
SpiderPlot(data, x = "x", group_by = "group")

# Use y values
data <- data.frame(
  x = rep(LETTERS[1:5], 2),
  y = c(1, 3, 6, 4, 2, 5, 7, 8, 9, 10),
  group = rep(c("G1", "G2"), each = 5)
)
RadarPlot(data, x = "x", y = "y", scale_y = "none", group_by = "group")
RadarPlot(data, x = "x", y = "y", facet_by = "group")
RadarPlot(data, x = "x", y = "y", split_by = "group")
RadarPlot(data,
  x = "x", y = "y", split_by = "group",
  palette = c(G1 = "Set1", G2 = "Paired")
)
```

RarefactionPlot *Rarefaction Plot*

Description

Generate rarefaction/extrapolation curves for diversity analysis using iNEXT. This function visualizes species diversity estimates with different levels of sampling effort.

Usage

```
RarefactionPlot(
  data,
  type = 1,
  se = NULL,
  group_by = "group",
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 0.2,
  pt_size = 3,
  line_width = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	An iNEXT object or a list of data that will be handled by iNEXT::iNEXT.
type	Three types of plots: sample-size-based rarefaction/extrapolation curve (type = 1); sample completeness curve (type = 2); coverage-based rarefaction/extrapolation curve (type = 3).
se	A logical variable to display confidence interval around the estimated sampling curve. Default to NULL which means TRUE if the data has the lower and upper bounds.
group_by	A character string indicating how to group the data (color the lines). Possible values are "q" and "group"
group_by_sep	A character string indicating how to separate the group_by column if both "q" and "group" are used. Default to "_".
group_name	A character string indicating the name of the group, showing as the legend title.
split_by	A character string indicating how to split the data and plots. Possible values are "q" and "group"
split_by_sep	Separator when concatenating multiple split_by columns
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
pt_size	A numeric value specifying the size of the points.
line_width	A numeric value specifying the width of the lines.
facet_by	A character string indicating how to facet the data and plots. Possible values are "q" and "group"
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row

axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
...	Additional arguments to pass to iNEXT::iNEXT when data is not an iNEXT object.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
spider <- list(
  Girdled = c(46, 22, 17, 15, 15, 9, 8, 6, 6, 4, rep(2, 4), rep(1, 12)),
  Logged = c(
    88, 22, 16, 15, 13, 10, 8, 8, 7, 7, 7, 5, 4, 4, 4, 3, 3, 3, 3,
    2, 2, 2, 2, rep(1, 14)
  )
)

RarefactionPlot(spider)
RarefactionPlot(spider, q = c(0, 1, 2), facet_by = "q")
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q")
RarefactionPlot(spider,
  q = c(0, 1, 2), split_by = "q",
  palette = c("0" = "Paired", "1" = "Set1", "2" = "Dark2")
)
RarefactionPlot(spider,
  q = c(0, 1, 2), group_by = "q",
  facet_by = "group", palette = "Set1", type = 3
)
```

RidgePlot *Ridge Plot*

Description

Creates ridge plots to illustrate the distribution of data across multiple groups. Groups are displayed on the y-axis with overlapping density curves.

Usage

```
RidgePlot(
  data,
  x = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
```

```
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale = NULL,
  add_vline = NULL,
  vline_type = "solid",
  vline_color = TRUE,
  vline_width = 0.5,
  vline_alpha = 1,
  flip = FALSE,
  alpha = 0.8,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  x_text_angle = 90,
  keep_empty = FALSE,
  reverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "none",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame (long or wide form)
x	Column name for values (numeric expected)
in_form	Data format: "long" or "wide"
split_by	Column to split data into multiple plots
split_by_sep	Separator for concatenating multiple split_by columns
group_by	Column(s) to group the data (shown on y-axis)
group_by_sep	Separator for concatenating multiple group_by columns

group_name	Legend title for group_by
scale	Scaling factor for ridges (higher = more overlap)
add_vline	Add vertical lines (TRUE for mean, numeric vector, or named list)
vline_type	Line type for vertical lines
vline_color	Color for vertical lines (TRUE to match groups)
vline_width	Width of vertical lines
vline_alpha	Alpha for vertical lines
flip	Whether to flip the plot
alpha	Transparency for ridges
theme	Theme name or function
theme_args	Arguments passed to theme function
palette	Palette name
palcolor	Custom colors
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
x_text_angle	Angle for x-axis text
keep_empty	Keep empty groups on y-axis
reverse	Reverse order of groups on y-axis
facet_by	Columns for faceting
facet_scales	Facet scales type
facet_ncol	Number of facet columns
facet_nrow	Number of facet rows
facet_byrow	Fill facets by row
aspect.ratio	Aspect ratio
legend.position	Legend position
legend.direction	Legend direction
combine	Whether to combine multiple plots
nrow	Number of rows for combined plots
ncol	Number of columns for combined plots
byrow	Fill combined plots by row
seed	Random seed
axes	Axis handling for combined plots
axis_titles	Axis title handling for combined plots
guides	Guide handling for combined plots
design	Custom design for combined plots
...	Additional arguments passed to geom_density_ridges

Value

A ggplot object, combined plot, or list of plots

Examples

```
set.seed(8525)
data <- data.frame(
  x = c(rnorm(250, -1), rnorm(250, 1)),
  group = rep(LETTERS[1:5], each = 100)
)
RidgePlot(data, x = "x") # fallback to density plot
RidgePlot(data, x = "x", add_vline = 0, vline_color = "black")
RidgePlot(data, x = "x", group_by = "group")
RidgePlot(data, x = "x", group_by = "group", reverse = TRUE)
RidgePlot(data, x = "x", group_by = "group", add_vline = TRUE, vline_color = TRUE)

# wide form
data_wide <- data.frame(
  A = rnorm(100),
  B = rnorm(100),
  C = rnorm(100),
  D = rnorm(100),
  E = rnorm(100),
  group = sample(letters[1:4], 100, replace = TRUE)
)
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide")
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", facet_by = "group")
```

RingPlot *Ring Plot*

Description

Creates ring plots to visualize hierarchical proportional data. A ring plot is similar to a pie chart but can display multiple concentric rings, making it useful for showing nested categorical data.

Usage

```
RingPlot(
  data,
  x = NULL,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  label = NULL,
  clockwise = TRUE,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
```

```
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame
x	Column name for the rings. If NULL, creates a single ring. Will be converted to factor.
y	Column name for values. If NULL, counts will be used.
group_by	Column name(s) for how each ring is divided. Multiple columns will be concatenated with group_by_sep.
group_by_sep	Separator for concatenating multiple group_by columns.
group_name	Name for the group_by variable in the legend.
label	Whether to show ring labels. NULL means auto (FALSE for 1 ring, TRUE for multiple).
clockwise	Whether to draw clockwise (default: TRUE).
split_by	Column(s) to split the data by, creating separate plots. Multiple columns will be concatenated with split_by_sep.
split_by_sep	Separator for concatenating multiple split_by columns.
facet_by	Column(s) to facet by (max 2 columns).
facet_scales	Scale type for facets: "fixed", "free", "free_x", or "free_y".
facet_ncol	Number of columns for faceting.
facet_nrow	Number of rows for faceting.
facet_byrow	Whether to fill facets by row.

theme	Theme name or function. Default is "theme_ggforge".
theme_args	List of arguments to pass to the theme function.
palette	Color palette name. Default is "Paired".
palcolor	Custom colors (overrides palette).
alpha	Transparency level (0-1). Default is 1.
aspect.ratio	Aspect ratio of the plot. Default is 1.
legend.position	Legend position ("none", "left", "right", "bottom", "top").
legend.direction	Legend direction ("horizontal" or "vertical").
title	Plot title. Can be a string or function.
subtitle	Plot subtitle.
xlab	X-axis label.
ylab	Y-axis label.
keep_empty	Keep empty factor levels.
combine	Whether to combine plots when split_by is used.
nrow	Number of rows when combining plots.
ncol	Number of columns when combining plots.
byrow	Whether to arrange plots by row when combining.
seed	Random seed for reproducibility. Default is 8525.
axes	Axis handling when combining plots.
axis_titles	Axis title handling when combining plots.
guides	Guide handling when combining plots.
design	Custom design for combining plots.
...	Additional arguments (currently unused).

Value

A ggplot object, patchwork object (if combine=TRUE), or list of plots

Examples

```
# Basic ring plot with single ring
RingPlot(datasets::iris, group_by = "Species")

# Multiple rings
data <- data.frame(
  x = c("A", "B", "C", "A", "B", "C"),
  y = c(1, 2, 3, 4, 5, 6),
  group = c("a", "a", "a", "b", "b", "b")
)
RingPlot(data, x = "x", y = "y", group_by = "group")

# With faceting
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", facet_by = "vs")

# With splitting and custom palettes
RingPlot(datasets::mtcars,
  x = "cyl", group_by = "carb", split_by = "vs",
  palette = c("0" = "Set1", "1" = "Paired")
)
```

ROCCurve *ROC Curve Plot*

Description

A wrapped function around plotROC package to create ROC curves. ROC (Receiver Operating Characteristic) curves are used to evaluate binary classification models by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity).

Usage

```
ROCCurve(
  data,
  truth_by,
  score_by,
  pos_label = NULL,
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_axis_reverse = FALSE,
  percent = FALSE,
  ci = NULL,
  n_cuts = 0,
  cutoffs_at = NULL,
  cutoffs_labels = NULL,
  cutoffs_accuracy = 0.001,
  cutoffs_pt_size = 5,
  cutoffs_pt_shape = 4,
  cutoffs_pt_stroke = 1,
  cutoffs_labal_fg = "black",
  cutoffs_label_size = 4,
  cutoffs_label_bg = "white",
  cutoffs_label_bg_r = 0.1,
  show_auc = c("auto", "none", "legend", "plot"),
  auc_accuracy = 0.01,
  auc_size = 4,
  increasing = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
```

```
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = ifelse(x_axis_reverse, "Specificity", "1 - Specificity"),
  ylab = "Sensitivity",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
truth_by	A character string of the column name that contains the true class labels. (a.k.a. the binary outcome, 1/0 or TRUE/FALSE.)
score_by	character strings of the column names that contains the predicted scores. When multiple columns are provided, the ROC curve is plotted for each column.
pos_label	A character string of the positive class label. When NULL, the labels will be handled by the plotROC package.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
group_by	Column name(s) for grouping data
group_by_sep	Separator when concatenating multiple group_by columns
group_name	A character string to name the legend of the ROC curve groups.
x_axis_reverse	A logical to reverse the x-axis, that is from 1 to 0.
percent	A logical to display the x and y axis as percentages.
ci	A list of arguments to pass to plotROC::geom_rocci() to add confidence intervals. When NULL, no confidence intervals are added.
n_cuts	An integer to specify the number of cutpoints on the ROC curve. It will be the quantiles of the predicted scores.
cutoffs_at	Vector of user supplied cutoffs to plot as points. If non-NULL, it will override the values of n_cuts and plot the observed cutoffs closest to the user-supplied ones. Both cutoffs_at and cutoffs.labels will be passed to plotROC::geom_roc(). Other than numeric values, the following special values are allowed. These values are the methods of OptimalCutpoints::optimal.cutpoints().
cutoffs_labels	vector of user-supplied labels for the cutoffs. Must be a character vector of the same length as cutoffs_at.
cutoffs_accuracy	A numeric to specify the accuracy of the cutoff values to show.
cutoffs_pt_size	A numeric to specify the size of the cutoff points.

cutoffs_pt_shape
: A numeric to specify the shape of the cutoff points.

cutoffs_pt_stroke
: A numeric to specify the stroke of the cutoff points.

cutoffs_labal_fg
: A character string to specify the color of the cutoff labels.

cutoffs_label_size
: A numeric to specify the size of the cutoff labels.

cutoffs_label_bg
: A character string to specify the background color of the cutoff labels.

cutoffs_label_bg_r
: A numeric to specify the radius of the background of the cutoff labels.

show_auc
: A character string to specify the position of the AUC values.
  - "auto" (default): Automatically determine the position based on the plot. When there is a single group or 'facet_by' is provided, the AUC is placed on the plot. Otherwise, the AUC is placed in the legend.
  - "none": Do not display the AUC values.
  - "legend": Display the AUC values in the legend.
  - "plot": Display the AUC values on the plot (left/right bottom corner).

auc_accuracy
: A numeric to specify the accuracy of the AUC values.

auc_size
: A numeric to specify the size of the AUC values when they are displayed on the plot.

increasing
: TRUE if the score is increasing with the truth (1), FALSE otherwise.

theme
: Theme name (string) or theme function

theme_args
: List of arguments passed to theme function

palette
: Color palette name

palcolor
: Custom colors for palette

alpha
: Transparency level (0-1)

facet_by
: Column name(s) for faceting the plot

facet_scales
: Scales for facets: "fixed", "free", "free_x", "free_y"

facet_ncol
: Number of columns in facet layout

facet_nrow
: Number of rows in facet layout

facet_byrow
: Fill facets by row (TRUE) or column (FALSE)

aspect.ratio
: Aspect ratio of plot panel

legend.position
: Legend position: "none", "left", "right", "bottom", "top"

legend.direction
: Legend direction: "horizontal" or "vertical"

title
: Plot title

subtitle
: Plot subtitle

xlab
: X-axis label

ylab
: Y-axis label

combine
: Whether to combine split plots into one

nrow
: Number of rows when combining plots

ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A patchwork::wrap_plots object or a list of them if combine is FALSE. You can retrieve the AUC values using attr(p, "auc") if combine is TRUE. If combine is FALSE, The AUC value of each plot can be retrieved using attr(p[[i]], "auc").

Examples

```
set.seed(8525)

D.ex <- rbinom(200, size = 1, prob = .5)
M1 <- rnorm(200, mean = D.ex, sd = .65)
M2 <- rnorm(200, mean = D.ex, sd = 1.5)
gender <- c("Male", "Female")[rbinom(200, 1, .49) + 1]

data <- data.frame(
  D = D.ex, D.str = c("Healthy", "Ill")[D.ex + 1],
  gender = gender, M1 = M1, M2 = M2
)

ROCCurve(data, truth_by = "D", score_by = "M1")
# will warn about the positive label
ROCCurve(data, truth_by = "D.str", score_by = "M1")
ROCCurve(data, truth_by = "D", score_by = "M1", increasing = FALSE)
# Multiple ROC curves
ROCCurve(data, truth_by = "D", score_by = c("M1", "M2"), group_name = "Method")
ROCCurve(data, truth_by = "D", score_by = "M1", group_by = "gender", show_auc = "plot")
# Reverse the x-axis and display the axes as percentages
ROCCurve(data, truth_by = "D", score_by = "M1", x_axis_reverse = TRUE, percent = TRUE)
# Pass additional arguments to geom_roc and make the curve black
ROCCurve(data, truth_by = "D", score_by = "M1", n_cuts = 10, palcolor = "black")
# Add confidence intervals
ROCCurve(data, truth_by = "D", score_by = "M1", ci = list(sig.level = .01))
# Facet by a column
ROCCurve(data, truth_by = "D", score_by = "M1", facet_by = "gender")
# Show cutoffs
ROCCurve(data, truth_by = "D", score_by = "M1", cutoffs_at = c(0, "ROC01", "SpEqualSe"))
# Split by a column
p <- ROCCurve(data, truth_by = "D", score_by = "M1", split_by = "gender")
p
# Retrieve the AUC values
attr(p, "auc")
# Retrieve the cutoffs
attr(p, "cutoffs")
```

SankeyPlot *Sankey / Alluvial Plot*

Description

A plot visualizing flow/movement/change from one state to another or one time to another. `AlluvialPlot` is an alias of `SankeyPlot`.

Usage

```
SankeyPlot(
  data,
  in_form = c("auto", "long", "lodes", "wide", "alluvia", "counts"),
  x,
  x_sep = "_",
  y = NULL,
  stratum = NULL,
  stratum_sep = "_",
  alluvium = NULL,
  alluvium_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  keep_empty = TRUE,
  flow = FALSE,
  expand = c(0, 0, 0, 0),
  nodes_legend = c("auto", "separate", "merge", "none"),
  nodes_color = "grey30",
  links_fill_by = NULL,
  links_fill_by_sep = "_",
  links_name = NULL,
  links_color = "gray80",
  nodes_palette = "Paired",
  nodes_palcolor = NULL,
  nodes_alpha = 1,
  nodes_label = FALSE,
  nodes_label_miny = 0,
  nodes_width = 0.25,
  links_palette = "Paired",
  links_palcolor = NULL,
  links_alpha = 0.6,
  legend.box = "vertical",
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  flip = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
```

```
  ylab = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

AlluvialPlot(
  data,
  in_form = c("auto", "long", "lodes", "wide", "alluvia", "counts"),
  x,
  x_sep = "_",
  y = NULL,
  stratum = NULL,
  stratum_sep = "_",
  alluvium = NULL,
  alluvium_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  keep_empty = TRUE,
  flow = FALSE,
  expand = c(0, 0, 0, 0),
  nodes_legend = c("auto", "separate", "merge", "none"),
  nodes_color = "grey30",
  links_fill_by = NULL,
  links_fill_by_sep = "_",
  links_name = NULL,
  links_color = "gray80",
  nodes_palette = "Paired",
  nodes_palcolor = NULL,
  nodes_alpha = 1,
  nodes_label = FALSE,
  nodes_label_miny = 0,
  nodes_width = 0.25,
  links_palette = "Paired",
  links_palcolor = NULL,
  links_alpha = 0.6,
  legend.box = "vertical",
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
```

```
  legend.direction = "vertical",
  flip = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame in following possible formats: • "long" or "lodes": A long format with columns for `x`, `stratum`, `alluvium`, and `y`. `x` (required, single columns or concatenated by `x_sep`) is the column name to plot on the x-axis, `stratum` (defaults to `links_fill_by`) is the column name to group the nodes for each `x`, `alluvium` (required) is the column name to define the links, and `y` is the frequency of each `x`, `stratum`, and `alluvium`. • "wide" or "alluvia": A wide format with columns for `x`. `x` (required, multiple columns, `x_sep` won't be used) are the columns to plot on the x-axis, `stratum` and `alluvium` will be ignored. See ggalluvial::to_lodes_form for more details. • "counts": A format with counts being provides under each `x`. `x` (required, multiple columns, `x_sep` won't be used) are the columns to plot on the x-axis. When the first element of `x` is ".", values of `links_fill_by` (required) will be added to the plot as the first column of nodes. It is useful to show how the links are flowed from the source to the targets. • "auto" (default): Automatically determine the format based on the columns provided. When the length of `x` is greater than 1 and all `x` columns are numeric, "counts" format will be used. When the length of `x` is greater than 1 and ggalluvial::is_alluvia_form returns TRUE, "alluvia" format will be used. Otherwise, "lodes" format will be tried.
in_form	A character string to specify the format of the data. Possible values are "auto", "long", "lodes", "wide", "alluvia", and "counts".
x	A character string of the column name to plot on the x-axis. See `data` for more details.

x_sep — A character string to concatenate the columns in x, if multiple columns are provided.

y — A character string of the column name to plot on the y-axis. When in_form is "counts", y will be ignored. Otherwise, it defaults to the count of each x, stratum, alluvium and links_fill_by.

stratum — A character string of the column name to group the nodes for each x. See data for more details.

stratum_sep — A character string to concatenate the columns in stratum, if multiple columns are provided.

alluvium — A character string of the column name to define the links. See data for more details.

alluvium_sep — A character string to concatenate the columns in alluvium, if multiple columns are provided.

split_by — Column name(s) to split data into multiple plots

split_by_sep — Separator when concatenating multiple split_by columns

keep_empty — A logical value to keep the empty nodes.

flow — A logical value to use ggalluvial::geom_flow instead of ggalluvial::geom_alluvium.

expand — Expansion values for axes (CSS-like: top, right, bottom, left)

nodes_legend — Controls how the legend of nodes will be shown. Possible values are:

- "merge": Merge the legends of nodes. That is only one legend will be shown for all nodes.
- "separate": Show the legends of nodes separately. That is, nodes on each x will have their own legend.
- "none": Do not show the legend of nodes.
- "auto": Automatically determine how to show the legend. When nodes_label is TRUE, "none" will apply. When nodes_label is FALSE, and if stratum is the same as links_fill_by, "none" will apply. If there is any overlapping values between the nodes on different x, "merge" will apply. Otherwise, "separate" will apply.

nodes_color — A character string to color the nodes. Use a special value ".fill" to use the same color as the fill.

links_fill_by — A character string of the column name to fill the links.

links_fill_by_sep — A character string to concatenate the columns in links_fill_by, if multiple columns are provided.

links_name — A character string to name the legend of links.

links_color — A character string to color the borders of links. Use a special value ".fill" to use the same color as the fill.

nodes_palette — A character string to specify the palette of nodes fill.

nodes_palcolor — A character vector to specify the colors of nodes fill.

nodes_alpha — A numeric value to specify the transparency of nodes fill.

nodes_label — A logical value to show the labels on the nodes.

nodes_label_miny — A numeric value to specify the minimum y (frequency) to show the labels.

nodes_width — A numeric value to specify the width of nodes.

links_palette	A character string to specify the palette of links fill.
links_palcolor	A character vector to specify the colors of links fill.
links_alpha	A numeric value to specify the transparency of links fill.
legend.box	A character string to specify the box of the legend, either "vertical" or "horizontal".
x_text_angle	Angle for x-axis text
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
flip	A logical value to flip the plot.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots
...	Other arguments to pass to ggalluvial::geom_alluvium or ggalluvial::geom_flow.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
# Reproduce the examples in ggalluvial
set.seed(8525)

data(UCBAdmissions, package = "datasets")
UCBAdmissions <- as.data.frame(UCBAdmissions)
SankeyPlot(as.data.frame(UCBAdmissions),
  x = c("Gender", "Dept"),
  y = "Freq", nodes_width = 1 / 12, links_fill_by = "Admit", nodes_label = TRUE,
  nodes_palette = "simspec", links_palette = "Set1", links_alpha = 0.5,
  nodes_palcolor = "black", links_color = "transparent"
)

data(HairEyeColor, package = "datasets")
SankeyPlot(as.data.frame(HairEyeColor),
  x = c("Hair", "Eye", "Sex"),
  y = "Freq", links_fill_by = "Eye", nodes_width = 1 / 8, nodes_alpha = 0.4,
  flip = TRUE, reverse = FALSE, knot.pos = 0, links_color = "transparent",
  ylab = "Freq", links_alpha = 0.5, links_name = "Eye (links)", links_palcolor = c(
    Brown = "#70493D", Hazel = "#E2AC76", Green = "#3F752B", Blue = "#81B0E4"
  )
)

data(Refugees, package = "alluvial")
country_regions <- c(
  Afghanistan = "Middle East",
  Burundi = "Central Africa",
  `Congo DRC` = "Central Africa",
  Iraq = "Middle East",
  Myanmar = "Southeast Asia",
  Palestine = "Middle East",
  Somalia = "Horn of Africa",
  Sudan = "Central Africa",
  Syria = "Middle East",
  Vietnam = "Southeast Asia"
)
Refugees$region <- country_regions[Refugees$country]
SankeyPlot(Refugees,
  x = "year", y = "refugees", alluvium = "country",
  links_fill_by = "country", links_color = ".fill", links_alpha = 0.75,
  links_palette = "Set3", facet_by = "region", x_text_angle = -45, nodes_legend = "none",
  theme_args = list(strip.background = ggplot2::element_rect(fill = "grey80")),
  decreasing = FALSE, nodes_width = 0, nodes_color = "transparent", ylab = "refugees",
  title = "Refugee volume by country and region of origin"
)

data(majors, package = "ggalluvial")
majors$curriculum <- as.factor(majors$curriculum)
SankeyPlot(majors,
  x = "semester", stratum = "curriculum", alluvium = "student",
  links_fill_by = "curriculum", flow = TRUE, stat = "alluvium", nodes_palette = "Set2",
  links_palette = "Set2"
)

data(vaccinations, package = "ggalluvial")
vaccinations <- transform(vaccinations,
```

```
  response = factor(response, rev(levels(response)))
)
SankeyPlot(vaccinations,
  x = "survey", stratum = "response", alluvium = "subject",
  y = "freq", links_fill_by = "response", nodes_label = TRUE, nodes_alpha = 0.5,
  nodes_palette = "seurat", links_palette = "seurat", links_alpha = 0.5,
  legend.position = "none", flow = TRUE, expand = c(0, 0, 0, .15), stat = "alluvium",
  title = "vaccination survey responses at three points in time"
)

data(Titanic, package = "datasets")
SankeyPlot(as.data.frame(Titanic),
  x = c("Class", "Sex"), y = "Freq",
  links_fill_by = "Survived", flow = TRUE, facet_by = "Age", facet_scales = "free_y",
  nodes_label = TRUE, expand = c(0.05, 0), xlab = "", links_palette = "Set1",
  nodes_palcolor = "white", nodes_label_miny = 10
)

# Simulated examples
df <- data.frame(
  Clone = paste0("clone", 1:10),
  Timepoint1 = sample(c(rep(0, 30), 1:100), 10),
  Timepoint2 = sample(c(rep(0, 30), 1:100), 10)
)
SankeyPlot(df,
  x = c("Timepoint1", "Timepoint2"), alluvium = "Clone",
  links_color = ".fill"
)

df <- data.frame(
  Clone = rep(paste0("clone", 1:6), each = 2),
  Timepoint1 = sample(c(rep(0, 30), 1:100), 6),
  Timepoint2 = sample(c(rep(0, 30), 1:100), 6),
  Group = rep(c("A", "B"), 6)
)
SankeyPlot(df,
  x = c(".", "Timepoint1", "Timepoint2"),
  stratum = "Group", links_fill_by = "Clone", links_color = ".fill"
)
```

`ScatterPlot` *Scatter Plot*

Description

Creates scatter plots with support for size mapping, color mapping, highlighting specific points, and custom transformations. Supports splitting by groups and faceting.

Usage

```
ScatterPlot(
  data,
  x,
```

```
  y,
  size_by = 2,
  size_name = NULL,
  color_by = NULL,
  color_name = NULL,
  color_reverse = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  shape = 21,
  alpha = NULL,
  border_color = "black",
  highlight = NULL,
  highlight_shape = 16,
  highlight_size = 3,
  highlight_color = "red",
  highlight_alpha = 1,
  xtrans = "identity",
  ytrans = "identity",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = NULL,
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for x-axis variable
y	Column name for y-axis variable
size_by	Column name to use for point size, or a numeric value for fixed size.

size_name	Name for the size legend.
color_by	Column name to use for point color. Can be numeric or categorical.
color_name	Name for the color legend.
color_reverse	Whether to reverse the color direction.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
shape	Shape of points (default 21 which has fill).
alpha	Transparency level (0-1)
border_color	Color for point borders, or TRUE to match fill color.
highlight	Vector of row indices/names or expression string to highlight points.
highlight_shape	Shape for highlighted points.
highlight_size	Size for highlighted points.
highlight_color	Color for highlighted points.
highlight_alpha	Alpha for highlighted points.
xtrans	Transformation for x-axis (e.g. "log10", "sqrt").
ytrans	Transformation for y-axis (e.g. "log10", "sqrt").
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility

axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, patchwork object (if combine=TRUE), or list of plots

Examples

```
# Create sample data
set.seed(8525)
data <- data.frame(
  x = rnorm(100),
  y = rnorm(100),
  size_val = abs(rnorm(100)),
  category = sample(c("A", "B", "C"), 100, replace = TRUE),
  group = sample(c("Group1", "Group2"), 100, replace = TRUE)
)

# Basic scatter plot
ScatterPlot(data, x = "x", y = "y")

# Highlight specific points
ScatterPlot(data, x = "x", y = "y", highlight = "x > 1 & y > 0")

# Size by numeric column
ScatterPlot(data, x = "x", y = "y", size_by = "size_val")

# Color by numeric column with continuous gradient
ScatterPlot(data, x = "x", y = "y", color_by = "size_val", palette = "RdYlBu")

# Color by categorical column
ScatterPlot(data, x = "x", y = "y", color_by = "category")

# Combine size and color mapping with custom border
ScatterPlot(data,
  x = "x", y = "y", size_by = "size_val", color_by = "category",
  border_color = "black"
)

# Border color matches fill color (for cohesive look)
ScatterPlot(data,
  x = "x", y = "y", color_by = "category",
  border_color = TRUE
)

# Use shape without fill (solid points)
ScatterPlot(data,
  x = "x", y = "y", color_by = "category",
  shape = 16, palette = "Set1"
)

# Split by group to create separate panels
```

```
ScatterPlot(data,
  x = "x", y = "y", color_by = "category",
  split_by = "group", combine = TRUE
)

# Facet by group (alternative to split_by)
ScatterPlot(data, x = "x", y = "y", color_by = "category", facet_by = "group")

# Log transformation of axes
data_pos <- data.frame(
  x = 10^rnorm(100, 2, 1),
  y = 10^rnorm(100, 3, 0.5)
)
ScatterPlot(data_pos, x = "x", y = "y", xtrans = "log10", ytrans = "log10")
```

`show_palettes` *Show available palettes*

Description

Display available color palettes visually

Usage

```
show_palettes(
  palettes = NULL,
  type = c("discrete", "continuous"),
  index = NULL,
  palette_names = NULL,
  return_names = TRUE,
  return_palettes = FALSE
)
```

Arguments

`palettes`	Custom palette list (NULL to use built-in)
`type`	Type of palettes to show: "discrete", "continuous", or both
`index`	Indices of palettes to show
`palette_names`	Specific palette names to show
`return_names`	Return palette names instead of plotting
`return_palettes`	Return palette colors instead of plotting

Value

Plot, palette names, or palette colors

spatialplots *Spatial Plotting Functions for ggforge*

Description

Functions for plotting spatial data including raster images, masks, shapes, and points. These functions work with `terra` SpatRaster and SpatVector objects, as well as regular data frames.

Usage

```
SpatImagePlot(
  data,
  ext = NULL,
  raster = NULL,
  raster_dpi = NULL,
  flip_y = TRUE,
  palette = "turbo",
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

SpatMasksPlot(
  data,
  ext = NULL,
  flip_y = TRUE,
  add_border = TRUE,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = "turbo",
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
```

```
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

SpatShapesPlot(
  data,
  x = NULL,
  y = NULL,
  group = NULL,
  ext = NULL,
  flip_y = TRUE,
  fill_by = NULL,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

## S3 method for class 'SpatVector'
SpatShapesPlot(
  data,
  x = NULL,
  y = NULL,
  group = NULL,
  ext = NULL,
  flip_y = TRUE,
```

```
  fill_by = NULL,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

## S3 method for class 'data.frame'
SpatShapesPlot(
  data,
  x,
  y,
  group,
  ext = NULL,
  flip_y = TRUE,
  fill_by = "grey90",
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
```

```
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

SpatPointsPlot(
  data,
  x = NULL,
  y = NULL,
  ext = NULL,
  flip_y = TRUE,
  color_by = NULL,
  size_by = NULL,
  size = NULL,
  fill_by = NULL,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  palette = NULL,
  palcolor = NULL,
  palette_reverse = FALSE,
  alpha = 1,
  color_name = NULL,
  size_name = NULL,
  shape = 16,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
  hex_binwidth = NULL,
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
```

```
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  label_insitu = FALSE,
  label_pos = c("median", "mean", "max", "min", "first", "last", "center", "random"),
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  graph = NULL,
  graph_x = NULL,
  graph_y = NULL,
  graph_xend = NULL,
  graph_yend = NULL,
  graph_value = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)
```

Arguments

data	A data.frame with spatial coordinates
ext	Spatial extent (SpatExtent or numeric vector of length 4)
raster	Whether to rasterize points
raster_dpi	Rasterization resolution
flip_y	Whether to flip y-axis
palette	Color palette name
palcolor	Custom colors
palette_reverse	Reverse palette
alpha	Transparency level
fill_name	Legend title

return_layer	Whether to return layers only
theme	Theme name or function
theme_args	Theme arguments
legend.position	Legend position
legend.direction	Legend direction
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed
add_border	Whether to add border around masks
border_color	Border color
border_size	Border width
border_alpha	Border transparency
x	X-coordinate column name
y	Y-coordinate column name
group	Grouping column (for data.frame)
fill_by	Alternative to color_by (for compatibility)
highlight	Rows to highlight
highlight_alpha	Highlight transparency
highlight_size	Highlight size
highlight_color	Highlight color
highlight_stroke	Highlight stroke width
facet_scales	Facet scales
facet_nrow	Number of facet rows
facet_ncol	Number of facet columns
facet_byrow	Fill facets by row
color_by	Column to color points by
size_by	Column to size points by
size	Fixed point size (alternative to size_by)
lower_quantile	Lower quantile for color scaling
upper_quantile	Upper quantile for color scaling
lower_cutoff	Lower cutoff for color values
upper_cutoff	Upper cutoff for color values
color_name	Legend title for color
size_name	Legend title for size
shape	Point shape

hex	Whether to use hex binning
hex_linewidth	Hex border width
hex_count	Whether to count in hex bins
hex_bins	Number of hex bins
hex_binwidth	Hex bin width
label	Whether to add labels
label_size	Label size
label_fg	Label foreground color
label_bg	Label background color
label_bg_r	Label background ratio
label_repel	Whether to repel labels
label_repulsion	Label repulsion force
label_pt_size	Label point size
label_pt_color	Label point color
label_segment_color	Label segment color
label_insitu	Whether to use actual labels in situ
label_pos	Label position function
graph	Graph/network adjacency matrix or data
graph_x	Graph x column
graph_y	Graph y column
graph_xend	Graph xend column
graph_yend	Graph yend column
graph_value	Graph value column
edge_size	Edge size range
edge_alpha	Edge transparency
edge_color	Edge color

Value

ggplot object or list of layers

ggplot object or list of layers

ggplot object or list of layers

ggplot object or list of layers

Examples

```
## Not run:
# Create a simple raster
library(terra)
r <- rast(ncols = 100, nrows = 100, xmin = 0, xmax = 100, ymin = 0, ymax = 100)
values(r) <- runif(ncell(r))

# Plot raster image
```

```
SpatImagePlot(r, palette = "viridis")

# Plot with custom extent
SpatImagePlot(r, ext = c(20, 80, 20, 80), palette = "turbo")

## End(Not run)
## Not run:
# Create a mask raster
library(terra)
r <- rast(ncols = 50, nrows = 50, xmin = 0, xmax = 50, ymin = 0, ymax = 50)
values(r) <- sample(0:3, ncell(r), replace = TRUE)

# Plot masks with borders
SpatMasksPlot(r, add_border = TRUE, palette = "Set2")

## End(Not run)
## Not run:
# Create polygon data
library(terra)
v <- vect("POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))")

# Plot shapes
SpatShapesPlot(v, fill_by = "grey80", border_color = "black")

# Plot with data.frame
poly_df <- data.frame(
  x = c(0, 10, 10, 0),
  y = c(0, 0, 10, 10),
  group = rep(1, 4)
)
SpatShapesPlot(poly_df, x = "x", y = "y", group = "group")

## End(Not run)
## Not run:
# Create spatial point data
spatial_data <- data.frame(
  x = runif(1000, 0, 100),
  y = runif(1000, 0, 100),
  cluster = sample(c("A", "B", "C"), 1000, replace = TRUE),
  value = rnorm(1000)
)

# Basic point plot
SpatPointsPlot(spatial_data, x = "x", y = "y", color_by = "cluster")

# Continuous color scale
SpatPointsPlot(spatial_data, x = "x", y = "y", color_by = "value", palette = "viridis")

# Hex binning for large datasets
SpatPointsPlot(spatial_data, x = "x", y = "y", color_by = "value", hex = TRUE)

# Add labels
SpatPointsPlot(spatial_data,
  x = "x", y = "y", color_by = "cluster",
  label = TRUE, label_repel = TRUE
)
```

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

SplitBarPlot *Split Bar Plot (Waterfall Plot)*

Description

Create a split bar plot showing positive and negative values on opposite sides of a central axis. Also known as a waterfall plot or diverging bar plot.

This is useful for showing data that has both positive and negative values, such as survey responses (agree/disagree), changes (increase/decrease), or any data with bidirectional nature.

Usage

```
SplitBarPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
 order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
```

```
  ylab = NULL,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

WaterfallPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
 order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column for x-axis values (numeric, positive and negative).
y	Column(s) for y-axis categories. Will be converted to factor.
y_sep	Separator for concatenating multiple y columns.
flip	Flip x and y axes.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
alpha_by	Column to use for transparency (alpha values).
alpha_reverse	Reverse the alpha scale.
alpha_name	Legend title for alpha.
order_y	Ordering specification for y-axis. List with "+" and "-" keys for positive/negative ordering, or "*" for overall. Values can be "x_asc", "x_desc", "alpha_asc", "alpha_desc".
bar_height	Height of bars (0-1).
lineheight	Line height for wrapped text labels.
max_charwidth	Maximum characters before wrapping y-axis labels.
fill_by	Column for fill colors (default: direction).
fill_by_sep	Separator for concatenating multiple fill columns.
fill_name	Legend title for fill.
direction_pos_name	Label for positive direction.
direction_neg_name	Label for negative direction.
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_nrow	Number of rows in facet layout

facet_ncol	Number of columns in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
aspect.ratio	Aspect ratio of plot panel
x_min	Minimum x-axis value.
x_max	Maximum x-axis value.
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
keep_empty	Keep empty factor levels
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, list of plots, or combined plots

Examples

```
# Basic split bar plot (diverging bars)
data <- data.frame(
  term = c("Item A", "Item B", "Item C", "Item D"),
  value = c(-10, 20, -15, 25),
  importance = c(1, 3, 2, 4)
)
SplitBarPlot(data, x = "value", y = "term")

# With alpha mapping to show importance
SplitBarPlot(data, x = "value", y = "term", alpha_by = "importance")

# With custom fill groups
data$category <- c("Type A", "Type A", "Type B", "Type B")
SplitBarPlot(data, x = "value", y = "term", fill_by = "category")

# Custom ordering (by value descending for positive, ascending for negative)
SplitBarPlot(
  data,
```

```
  x = "value", y = "term",
  order_y = list("+" = "x_desc", "-" = "x_asc")
)

# Survey-style diverging bars (e.g., agree/disagree)
survey_data <- data.frame(
  question = rep(c(
    "Q1: Service quality", "Q2: Value for money",
    "Q3: Overall satisfaction"
  ), 2),
  response = c(45, 60, 55, -20, -15, -10),
  group = rep(c("Positive", "Negative"), each = 3)
)
SplitBarPlot(
  survey_data,
  x = "response", y = "question",
  direction_pos_name = "Agree",
  direction_neg_name = "Disagree",
  palette = "RdBu"
)
```

`themes` *ggforge Theming System*

Description

A flexible and elegant theming system for ggforge plots

`theme_ggforge` *Main ggforge Theme*

Description

The default theme for ggforge, providing a clean and modern appearance

Usage

```
theme_ggforge(aspect.ratio = NULL, base_size = NULL, font_family = NULL, ...)
```

Arguments

`aspect.ratio`	Aspect ratio of the plot panel
`base_size`	Base font size (scales all text elements)
`font_family`	Font family for all text
`...`	Additional arguments passed to `theme`

Value

A ggplot2 theme object

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_ggforge()
```

`theme_ggforge_grid` *ggforge Theme for Grid-based Plots (Heatmap-like)*

Description

A theme based on theme_bw, designed for grid-based plots like heatmaps, dot plots, and correlation matrices

Usage

```
theme_ggforge_grid(
  aspect.ratio = NULL,
  base_size = NULL,
  font_family = NULL,
  ...
)
```

Arguments

`aspect.ratio`	Aspect ratio of the plot panel
`base_size`	Base font size (scales all text elements)
`font_family`	Font family for all text
`...`	Additional arguments passed to `theme`

Value

A ggplot2 theme object

`theme_minimal_axes` *Minimal Theme*

Description

A minimal theme with coordinate axes

Usage

```
theme_minimal_axes(
  add_coord = TRUE,
  xlen_npc = 0.15,
  ylen_npc = 0.15,
  xlab = "",
  ylab = "",
  lab_size = 12,
  ...
)
```

Arguments

add_coord	Whether to add coordinate arrows
xlen_npc	Length of x-axis arrow (in npc units)
ylen_npc	Length of y-axis arrow (in npc units)
xlab	X-axis label
ylab	Y-axis label
lab_size	Label size
...	Additional arguments passed to theme

Value

List of ggplot2 theme components

TrendPlot *Trend Plot*

Description

Creates trend plots that combine area and bar visualizations to show trends. Like an area plot but with gaps between the bars. Supports splitting by groups, faceting, and custom color palettes.

Usage

```
TrendPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = ggplot2::waiver(),
  legend.direction = "vertical",
  title = NULL,
```

```
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column name for x-axis variable
y	Column name for y-axis variable
x_sep	Separator for concatenating multiple x columns.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
group_by	Column name(s) for grouping data
group_by_sep	Separator when concatenating multiple group_by columns
group_name	Name for the group legend.
scale_y	Whether to scale y-axis to proportions (0-1).
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
x_text_angle	Angle for x-axis text labels
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle

xlab	X-axis label
ylab	Y-axis label
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, patchwork object (if combine=TRUE), or list of plots

See Also

AreaPlot

Examples

```
# Create sample data
data <- data.frame(
  x = rep(c("A", "B", "C", "D"), 2),
  y = c(1, 3, 6, 4, 2, 5, 7, 8),
  group = rep(c("F1", "F2"), each = 4)
)

# Basic trend plot
TrendPlot(data, x = "x", y = "y", group_by = "group")

# With scaled y-axis
TrendPlot(data, x = "x", y = "y", group_by = "group", scale_y = TRUE)

# Split by group
TrendPlot(data, x = "x", y = "y", split_by = "group")

# Custom palettes per split
TrendPlot(data,
  x = "x", y = "y", split_by = "group",
  palette = list(F1 = "Set1", F2 = "Paired")
)
```

UpsetPlot *Upset Plot*

Description

Creates an UpSet plot to visualize intersections between sets. UpSet plots are a more scalable alternative to Venn diagrams for showing set intersections. The function supports multiple input formats: list, long, wide, and pre-processed UpsetPlotData objects.

Usage

```
UpsetPlot(
  data,
  in_form = c("auto", "long", "wide", "list", "upset"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = TRUE,
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  palette = "material-indigo",
  palcolor = NULL,
  alpha = 1,
  specific = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 0.6,
  legend.position = "right",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot

in_form	A character string indicating the datatype of the input data. Possible values are "auto", "long", "wide", "list", or "upset". Default is "auto" which will detect the format automatically.
split_by	Column to split the data by for creating multiple plots
split_by_sep	Separator for concatenating split_by columns
group_by	Column(s) to group by (depends on in_form)
group_by_sep	Separator for concatenating group_by columns
id_by	Column identifying unique instances (required for long format)
label	Whether to show count labels on bars
label_fg	Color of label text
label_size	Size of label text
label_bg	Background color of labels
label_bg_r	Radius of label background
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
specific	Show only specific intersections (not all overlaps)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
combine	Whether to combine multiple plots
nrow	Number of rows for combined plots
ncol	Number of columns for combined plots
byrow	Fill plots by row
seed	Random seed for reproducibility
axes	Axis handling for combined plots
axis_titles	Axis title handling for combined plots
guides	Guide handling for combined plots
design	Custom design for combined plots
...	Additional arguments passed to ggupset::scale_x_upset

Value

A ggplot object, wrap_plots object, or a list of ggplot objects

Examples

```
# Example 1: Basic upset plot with list input
data <- list(
  A = 1:5,
  B = 2:6,
  C = 3:7,
  D = 4:8
)
UpsetPlot(data)

# Example 2: Without labels
UpsetPlot(data, label = FALSE)

# Example 3: Custom palette and show all overlaps
UpsetPlot(data, palette = "Reds", specific = FALSE)

# Example 4: Limit number of sets and intersections
UpsetPlot(data, n_sets = 3, n_intersections = 10)

# Example 5: Wide format data
wide_data <- data.frame(
  A = c(TRUE, TRUE, FALSE, TRUE),
  B = c(TRUE, FALSE, TRUE, TRUE),
  C = c(FALSE, TRUE, TRUE, TRUE)
)
UpsetPlot(wide_data, in_form = "wide")
```

---

var_type *Annotate variable with explicit type*

Description

Allows users to override automatic type detection.

Usage

```
var_type(x, type = c("continuous", "discrete", "temporal", "ordered"))
```

Arguments

x	Vector. The variable to annotate
type	Character. One of "continuous", "discrete", "temporal", "ordered"

Value

The vector with type attribute attached

Examples

```
## Not run:
# Force a numeric year column to be treated as discrete
data$year <- var_type(data$year, "discrete")
```

```
# Force a character column to be treated as continuous (rare)
data$score <- var_type(as.numeric(data$score), "continuous")

## End(Not run)
```

`VelocityPlot` *Cell Velocity Plot*

Description

Visualize RNA velocity vectors on cell embeddings using arrows or streamlines. Supports three visualization modes: raw (cell-level arrows), grid (smoothed arrows), and stream (continuous streamlines).

Usage

```
VelocityPlot(
  embedding,
  v_embedding,
  plot_type = c("raw", "grid", "stream"),
  group_by = NULL,
  group_name = "Group",
  group_palette = "Paired",
  group_palcolor = NULL,
  n_neighbors = NULL,
  density = 1,
  smooth = 0.5,
  scale = 1,
  min_mass = 1,
  cutoff_perc = 5,
  arrow_angle = 20,
  arrow_color = "black",
  arrow_alpha = 1,
  streamline_l = 5,
  streamline_minl = 1,
  streamline_res = 1,
  streamline_n = 15,
  streamline_width = c(0, 0.8),
  streamline_alpha = 1,
  streamline_color = NULL,
  streamline_palette = "RdYlBu",
  streamline_palcolor = NULL,
  streamline_bg_color = "white",
  streamline_bg_stroke = 0.5,
  aspect.ratio = 1,
  title = "Cell velocity",
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  theme = "theme_ggforge",
  theme_args = list(),
  return_layer = FALSE,
  seed = 8525
)
```

Arguments

embedding	Matrix or data.frame of dimension n_obs x n_dim with cell embedding coordinates
v_embedding	Matrix or data.frame of dimension n_obs x n_dim with velocity vectors
plot_type	Type of plot: "raw" (cell-level arrows), "grid" (smoothed grid arrows), or "stream" (streamlines)
group_by	Optional grouping variable for coloring arrows (only for plot_type = "raw")
group_name	Name for the grouping legend
group_palette	Palette for group colors
group_palcolor	Custom colors for groups (overrides group_palette)
n_neighbors	Number of nearest neighbors for grid computation
density	Density of grid points (or sampling density for raw mode between 0-1)
smooth	Smoothing factor for velocity vectors
scale	Scaling factor for velocity vectors
min_mass	Minimum mass for grid points
cutoff_perc	Percentile cutoff for low-density grid points
arrow_angle	Angle of arrowheads in degrees
arrow_color	Color of arrow heads
arrow_alpha	Transparency of arrows (raw and grid modes)
streamline_l	Length of streamlines
streamline_minl	Minimum length of streamlines
streamline_res	Resolution of streamlines
streamline_n	Number of streamlines
streamline_width	Width range for streamlines
streamline_alpha	Transparency of streamlines
streamline_color	Fixed color for streamlines (NULL for gradient)
streamline_palette	Palette for streamline gradient
streamline_palcolor	Custom colors for streamline gradient
streamline_bg_color	Background color for streamlines
streamline_bg_stroke	Background stroke width

aspect.ratio	Aspect ratio of plot
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
legend.position	Legend position
legend.direction	Legend direction
theme	Theme name or function
theme_args	List of arguments passed to theme function
return_layer	Return ggplot layer instead of complete plot
seed	Random seed

Value

A ggplot object or ggplot layer if return_layer = TRUE

Examples

```
# Load example data
data(dim_example)

# Basic raw velocity plot
VelocityPlot(
  embedding = dim_example[, 1:2],
  v_embedding = dim_example[, 3:4]
)

# Velocity plot with grouping
VelocityPlot(
  embedding = dim_example[, 1:2],
  v_embedding = dim_example[, 3:4],
  group_by = dim_example$clusters,
  group_palette = "Set2"
)

# Grid-based velocity plot (smoothed arrows)
VelocityPlot(
  embedding = dim_example[, 1:2],
  v_embedding = dim_example[, 3:4],
  plot_type = "grid",
  density = 2
)

# Streamline velocity plot
VelocityPlot(
  embedding = dim_example[, 1:2],
  v_embedding = dim_example[, 3:4],
  plot_type = "stream",
  streamline_n = 20
)
```

VennDiagram *Venn Diagram*

Description

Create Venn diagrams to visualize overlaps between sets. Supports multiple input formats (long, wide, list) and various styling options.

Usage

```
VennDiagram(
  data,
  in_form = c("auto", "long", "wide", "list", "venn"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = "count",
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  fill_mode = "count",
  fill_name = NULL,
  palette = ifelse(fill_mode == "set", "Paired", "Spectral"),
  palcolor = NULL,
  alpha = 1,
  theme = "theme_ggforge",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
in_form	Format of input data. One of:

- "auto": Automatically detect format (default)
- "long": Long format with group_by and id_by columns
- "wide": Wide format with logical/0-1 columns
- "list": Named list of character vectors
- "venn": Pre-processed VennPlotData object

`split_by` Column name(s) to split data into multiple plots

`split_by_sep` Separator when concatenating multiple split_by columns

`group_by` Column(s) for grouping when in_form is "long" or "wide"

`group_by_sep` Separator for concatenating multiple group_by columns

`id_by` Column containing IDs when in_form is "long"

`label` Label style: "count", "percent", "both", "none", or a custom function

`label_fg` Color of label text

`label_size` Size of label text (default scaled by base_size)

`label_bg` Background color of labels

`label_bg_r` Radius of label background

`fill_mode` Fill coloring mode:

- "count": Color by intersection count (continuous)
- "set": Color by set membership (discrete, blended)
- "count_rev": Reverse of count mode

`fill_name` Name for the fill legend

`palette` Color palette name

`palcolor` Custom colors for palette

`alpha` Transparency level (0-1)

`theme` Theme name (string) or theme function

`theme_args` List of arguments passed to theme function

`title` Plot title

`subtitle` Plot subtitle

`legend.position` Legend position: "none", "left", "right", "bottom", "top"

`legend.direction` Legend direction: "horizontal" or "vertical"

`combine` Whether to combine split plots into one

`nrow` Number of rows when combining plots

`ncol` Number of columns when combining plots

`byrow` Fill combined plots by row

`seed` Random seed for reproducibility

`axes` How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")

`axis_titles` How to handle axis titles in combined plots

`guides` How to handle guides in combined plots ("collect", "keep", "auto")

`design` Custom layout design for combined plots

Value

A ggplot object or list/combined plots

Examples

```
set.seed(8525)
data <- list(
  A = sort(sample(letters, 8)),
  B = sort(sample(letters, 8)),
  C = sort(sample(letters, 8)),
  D = sort(sample(letters, 8))
)

VennDiagram(data)
VennDiagram(data, fill_mode = "set")
VennDiagram(data, label = "both")
VennDiagram(data, palette = "material-indigo", alpha = 0.6)
```

ViolinPlot *Violin Plot*

Description

Create violin plots with optional grouping, faceting, and statistical comparisons. Violin plots combine box plots with kernel density estimation to show the distribution shape of continuous data.

Usage

```
ViolinPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
 sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  keep_empty = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = NULL,
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  fill_reverse = FALSE,
  theme = "theme_ggforge",
  theme_args = list(),
```

```
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  add_point = FALSE,
  pt_color = "grey30",
  pt_size = NULL,
  pt_alpha = 1,
  jitter_width = NULL,
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  add_box = FALSE,
  box_color = "black",
  box_width = 0.1,
  box_ptsize = 2.5,
  add_trend = FALSE,
  trend_color = NULL,
  trend_linewidth = 1,
  trend_ptsize = 2,
  add_stat = NULL,
  stat_name = NULL,
  stat_color = "black",
  stat_size = 1,
  stat_stroke = 1,
  stat_shape = 25,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  comparisons = NULL,
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = "p.format",
  sig_labelsize = 3.5,
  hide_ns = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
```

```
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	Column for x-axis (discrete). Can be a single column name or multiple columns that will be concatenated.
y	Column for y-axis (numeric). The response variable.
in_form	Input data form: "long" (default) or "wide"
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
symnum_args	Symbolic number coding arguments for significance
sort_x	Sort x-axis values: "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median"
flip	Logical; flip coordinates to create horizontal plots
keep_empty	Logical; keep empty factor levels on x-axis
group_by	Column for grouping (creates dodged/side-by-side plots)
group_by_sep	Separator when concatenating multiple group_by columns
group_name	Legend name for groups
paired_by	Column identifying paired observations (for paired tests)
x_text_angle	Angle for x-axis text labels
step_increase	Step increase for comparison brackets
fill_mode	Fill coloring mode: "dodge", "x", "mean", or "median"
fill_reverse	Logical; reverse gradient fills
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
aspect.ratio	Aspect ratio of plot panel

`legend.position`
: Legend position: "none", "left", "right", "bottom", "top"

`legend.direction`
: Legend direction: "horizontal" or "vertical"

`add_point`
: Logical; add jittered data points

`pt_color`
: Point color (default: "grey30")

`pt_size`
: Point size (auto-calculated if NULL)

`pt_alpha`
: Point transparency (0-1)

`jitter_width`
: Jitter width for points

`jitter_height`
: Jitter height for points

`stack`
: Logical; stack facets vertically/horizontally

`y_max`
: Y-axis maximum (numeric or "qXX" for quantile)

`y_min`
: Y-axis minimum (numeric or "qXX" for quantile)

`add_box`
: Logical; add box overlay (violin only)

`box_color`
: Box overlay color

`box_width`
: Box overlay width

`box_ptsize`
: Box median point size

`add_trend`
: Logical; add trend line connecting medians

`trend_color`
: Trend line color

`trend_linewidth`
: Trend line width

`trend_ptsize`
: Trend point size

`add_stat`
: Function to add stat summary (e.g., mean)

`stat_name`
: Stat legend name

`stat_color`
: Stat point color

`stat_size`
: Stat point size

`stat_stroke`
: Stat point stroke width

`stat_shape`
: Stat point shape

`add_bg`
: Logical; add alternating background

`bg_palette`
: Background color palette

`bg_palcolor`
: Background custom colors

`bg_alpha`
: Background transparency

`add_line`
: Numeric; add horizontal reference line at this value

`line_color`
: Reference line color

`line_width`
: Reference line width

`line_type`
: Reference line type

`highlight`
: Points to highlight (logical, indices, or expression)

`highlight_color`
: Highlight color

`highlight_size`
: Highlight size

`highlight_alpha`
: Highlight transparency

comparisons	Pairwise comparisons (list of pairs or TRUE for all)
ref_group	Reference group for comparisons
pairwise_method	Statistical method for pairwise comparisons
multiplegroup_comparisons	Logical; perform multiple group comparisons
multiple_method	Statistical method for multiple comparisons
sig_label	Significance label format: "p.format" or "p.signif"
sig_labelsize	Significance label font size
hide_ns	Logical; hide non-significant comparisons
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")

Value

A ggplot object or combined plots (patchwork)

Examples

```
# ============================================================
# Basic Examples
# ============================================================

# Create sample data with different distributions
set.seed(456)
data <- data.frame(
  category = rep(c("Normal", "Bimodal", "Skewed", "Uniform"), each = 100),
  value = c(
    rnorm(100, 50, 10), # Normal
    c(rnorm(50, 30, 5), rnorm(50, 70, 5)), # Bimodal
```

```
    rexp(100, 0.1), # Skewed
    runif(100, 20, 80) # Uniform
  ),
  group = rep(c("A", "B"), 200)
)

# Simple violin plot
ViolinPlot(data, x = "category", y = "value")

# With title and labels
ViolinPlot(data,
  x = "category", y = "value",
  title = "Distribution Shapes Comparison",
  xlab = "Distribution Type",
  ylab = "Value"
)

# ============================================================
# Violin with Box Plot Overlay
# ============================================================

# Add box plot inside violin (shows quartiles)
ViolinPlot(data,
  x = "category", y = "value",
  add_box = TRUE
)

# Customize box overlay
ViolinPlot(data,
  x = "category", y = "value",
  add_box = TRUE,
  box_color = "white",
  box_width = 0.15,
  box_ptsize = 3
)

# ============================================================
# Adding Data Points
# ============================================================

# Add jittered points to show individual observations
ViolinPlot(data,
  x = "category", y = "value",
  add_point = TRUE
)

# Combine box overlay with points
ViolinPlot(data,
  x = "category", y = "value",
  add_box = TRUE,
  add_point = TRUE,
  pt_alpha = 0.3
)

# Customize point appearance
ViolinPlot(data,
  x = "category", y = "value",
```

```
  add_point = TRUE,
  pt_color = "navy",
  pt_size = 0.8,
  pt_alpha = 0.5,
  jitter_width = 0.3
)

# ============================================================
# Grouped Violin Plots
# ============================================================

# Side-by-side violins by group
ViolinPlot(data,
  x = "category", y = "value",
  group_by = "group"
)

# With box overlay
ViolinPlot(data,
  x = "category", y = "value",
  group_by = "group",
  add_box = TRUE
)

# Custom palette
ViolinPlot(data,
  x = "category", y = "value",
  group_by = "group",
  palette = "Set2",
  add_box = TRUE
)

# ============================================================
# Statistical Comparisons
# ============================================================

# Compare specific distributions
ViolinPlot(data,
  x = "category", y = "value",
  comparisons = list(
    c("Normal", "Bimodal"),
    c("Normal", "Skewed")
  )
)

# All pairwise comparisons
ViolinPlot(data,
  x = "category", y = "value",
  comparisons = TRUE,
  sig_label = "p.signif"
)

# Grouped comparisons
ViolinPlot(data,
  x = "category", y = "value",
  group_by = "group",
  comparisons = TRUE
```

```
)

# ============================================================
# Paired Data Analysis
# ============================================================

# Create paired data
paired_data <- data.frame(
  condition = factor(rep(c("Baseline", "Treatment"), each = 30)),
  patient = factor(rep(1:30, 2)),
  response = c(rnorm(30, 100, 20), rnorm(30, 120, 20))
)

# Paired violin with connecting lines
ViolinPlot(paired_data,
  x = "condition", y = "response",
  paired_by = "patient",
  add_box = TRUE
)

# With paired t-test
ViolinPlot(paired_data,
  x = "condition", y = "response",
  paired_by = "patient",
  comparisons = list(c("Baseline", "Treatment")),
  pairwise_method = "t.test"
)

# ============================================================
# Highlighting and Visual Enhancements
# ============================================================

# Highlight extreme values
ViolinPlot(data,
  x = "category", y = "value",
  add_point = TRUE,
  highlight = "value > 80 | value < 10",
  highlight_color = "red",
  highlight_size = 2
)

# Add trend line
ViolinPlot(data,
  x = "category", y = "value",
  add_trend = TRUE,
  trend_linewidth = 1.5
)

# Add reference line
ViolinPlot(data,
  x = "category", y = "value",
  add_line = 50,
  line_color = "darkgreen",
  line_type = 1
)

# Add mean indicator
```

```
ViolinPlot(data,
  x = "category", y = "value",
  add_stat = mean,
  stat_name = "Mean",
  stat_color = "red",
  stat_shape = 18,
  stat_size = 3
)

# ============================================================
# Fill Modes
# ============================================================

# Fill by x category
ViolinPlot(data,
  x = "category", y = "value",
  fill_mode = "x",
  palette = "Pastel1"
)

# Fill by mean (gradient coloring)
ViolinPlot(data,
  x = "category", y = "value",
  fill_mode = "mean",
  palette = "RdYlGn"
)

# Fill by median with reversed gradient
ViolinPlot(data,
  x = "category", y = "value",
  fill_mode = "median",
  palette = "Blues",
  fill_reverse = TRUE
)

# ============================================================
# Sorting and Orientation
# ============================================================

# Sort by mean value
ViolinPlot(data,
  x = "category", y = "value",
  sort_x = "mean_desc",
  add_box = TRUE
)

# Horizontal violin plot
ViolinPlot(data,
  x = "category", y = "value",
  flip = TRUE,
  add_box = TRUE
)

# ============================================================
# Faceting
# ============================================================
```

```
# Add faceting variable
data$experiment <- sample(c("Exp1", "Exp2"), nrow(data), replace = TRUE)

# Facet by experiment
ViolinPlot(data,
  x = "category", y = "value",
  facet_by = "experiment",
  add_box = TRUE
)

# Free scales
ViolinPlot(data,
  x = "category", y = "value",
  facet_by = "experiment",
  facet_scales = "free_y"
)

# ============================================================
# Wide Format Data
# ============================================================

# Wide format input
wide_data <- data.frame(
  Control = rnorm(50, 100, 15),
  LowDose = rnorm(50, 110, 15),
  HighDose = rnorm(50, 130, 20)
)

ViolinPlot(wide_data,
  x = c("Control", "LowDose", "HighDose"),
  in_form = "wide",
  add_box = TRUE,
  xlab = "Treatment",
  ylab = "Response"
)

# ============================================================
# Complex Example: Publication-Ready Plot
# ============================================================

# Gene expression data example
expr_data <- data.frame(
  gene = rep(c("BRCA1", "TP53", "EGFR", "MYC"), each = 40),
  expression = c(
    rnorm(40, 8, 1.5),
    rnorm(40, 6, 2),
    rnorm(40, 10, 1),
    rnorm(40, 12, 2.5)
  ),
  tissue = rep(rep(c("Normal", "Tumor"), each = 20), 4)
)

ViolinPlot(expr_data,
  x = "gene",
  y = "expression",
  group_by = "tissue",
  add_box = TRUE,
```

```
  add_point = TRUE,
  pt_alpha = 0.4,
  comparisons = TRUE,
  sig_label = "p.signif",
  hide_ns = TRUE,
  palette = c("Normal" = "#4DAF4A", "Tumor" = "#E41A1C"),
  title = "Gene Expression by Tissue Type",
  xlab = "Gene",
  ylab = "Expression (log2)",
  legend.position = "bottom"
)
```

VolcanoPlot *Volcano Plot*

Description

A volcano plot is a type of scatter plot that shows statistical significance (usually on the y-axis) versus magnitude of change (usually on the x-axis).

Usage

```
VolcanoPlot(
  data,
  x,
  y,
  ytrans = function(n) -log10(n),
  color_by = NULL,
  color_name = NULL,
  flip_negatives = FALSE,
  x_cutoff = NULL,
  y_cutoff = 0.05,
  trim = c(0, 1),
  xlim = NULL,
  x_cutoff_name = NULL,
  y_cutoff_name = NULL,
  x_cutoff_color = "red2",
  y_cutoff_color = "blue2",
  x_cutoff_linetype = "dashed",
  y_cutoff_linetype = "dashed",
  x_cutoff_linewidth = 0.5,
  y_cutoff_linewidth = 0.5,
  pt_size = 2,
  pt_alpha = 0.5,
  nlabel = 5,
  labels = NULL,
  label_by = NULL,
  label_size = 3,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
```

```
  highlight = NULL,
  highlight_color = "red",
  highlight_size = 2,
  highlight_alpha = 1,
  highlight_stroke = 0.5,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = NULL,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
x	A character string of the column name to plot on the x-axis (e.g., log fold change).
y	A character string of the column name to plot on the y-axis (e.g., p-value).
ytrans	A function to transform the y-axis values. Default is `-log10(n)`.
color_by	A character vector of column names to color the points by. If NULL, the points will be filled by the x and y cutoff value.
color_name	A character string to name the legend of color.
flip_negatives	A logical value to flip the y-axis for negative x values.
x_cutoff	A numeric value to set the x-axis cutoff. Both negative and positive of this value will be used.
y_cutoff	A numeric value to set the y-axis cutoff. Note that the y-axis cutoff will be transformed by `ytrans`.

trim	A numeric vector of length 2 to trim the x-axis values. The values must be in the range from 0 to 1, which works as quantile to trim the x-axis values. For example, c(0.01, 0.99) will trim the 1% and 99% quantile of the x-axis values.
xlim	A numeric vector of length 2 to set the x-axis limits.
x_cutoff_name	A character string to name the x-axis cutoff. If "none", the legend for the x-axis cutoff will not be shown.
y_cutoff_name	A character string to name the y-axis cutoff. If "none", the legend for the y-axis cutoff will not be shown.
x_cutoff_color	A character string to color the x-axis cutoff line.
y_cutoff_color	A character string to color the y-axis cutoff line.
x_cutoff_linetype	A character string to set the x-axis cutoff line type.
y_cutoff_linetype	A character string to set the y-axis cutoff line type.
x_cutoff_linewidth	A numeric value to set the x-axis cutoff line size.
y_cutoff_linewidth	A numeric value to set the y-axis cutoff line size.
pt_size	A numeric value to set the point size.
pt_alpha	A numeric value to set the point transparency.
nlabel	A numeric value to set the number of labels to show. The points will be ordered by the distance to the origin. Top nlabel points will be labeled.
labels	A character vector of row names or indexes to label the points.
label_by	A character string of column name to use as labels. If NULL, the row names will be used.
label_size	A numeric value to set the label size.
label_fg	A character string to set the label color.
label_bg	A character string to set the label background color.
label_bg_r	A numeric value specifying the radius of the background of the label.
highlight	A character vector of row names or indexes to highlight the points.
highlight_color	A character string to set the highlight color.
highlight_size	A numeric value to set the highlight size.
highlight_alpha	A numeric value to set the highlight transparency.
highlight_stroke	A numeric value to set the highlight stroke size.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)

theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function
palette	Color palette name
palcolor	Custom colors for palette
title	Plot title
subtitle	Plot subtitle
xlab	X-axis label
ylab	Y-axis label
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
seed	Random seed for reproducibility
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
# Obtained by Seurat::FindMarkers for the first cluster of pbmc_small
data <- data.frame(
  avg_log2FC = c(
    -3.69, -4.10, -2.68, -3.51, -3.09, -2.52, -3.53, -3.35, -2.82, -2.71,
    3.67, 4.79, 10.14, 5.36, 4.56, 4.62, 3.31, 4.72, 3.01, 3.86
  ),
  p_val_adj = c(
    3.82e-09, 1.52e-07, 1.79e-07, 4.68e-07, 4.83e-07, 6.26e-07, 2.61e-06,
    1.33e-05, 1.79e-05, 3.71e-05, 8.93e-04, 9.61e-04, 1.47e-03, 4.35e-03,
    4.85e-03, 5.12e-03, 1.90e-02, 2.13e-02, 3.80e-02, 6.72e-02
  ),
  gene = c(
    "HLA-DPB1", "LYZ", "HLA-DRA", "TYMP", "HLA-DPA1", "HLA-DRB1", "CST3",
    "HLA-DQB1", "HLA-DRB5", "LST1", "CCL5", "LCK", "MS4A6A", "CD3D", "CD7",
    "CD3E", "CTSW", "GZMM", "GZMA", "IL32"
  ),
  group = sample(LETTERS[1:2], 20, replace = TRUE)
)
```

```
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "-log10(0.05)")
VolcanoPlot(data,
  x = "avg_log2FC", y = "p_val_adj",
  label_by = "gene", y_cutoff_name = "-log10(0.05)"
)
VolcanoPlot(data,
  x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
  flip_negatives = TRUE, label_by = "gene"
)
VolcanoPlot(data,
  x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
  flip_negatives = TRUE, facet_by = "group", label_by = "gene"
)
VolcanoPlot(data,
  x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
  flip_negatives = TRUE, split_by = "group", label_by = "gene"
)
```

`WordCloudPlot` *Word Cloud Plot*

Description

Creates word cloud plots to illustrate word count/frequency. Words can be sized by count and colored by score. Supports splitting by groups, faceting, and custom color palettes.

Usage

```
WordCloudPlot(
  data,
  word_by = NULL,
  sentence_by = NULL,
  count_by = NULL,
  score_by = NULL,
  count_name = NULL,
  score_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  words_excluded = ggforge::words_excluded,
  score_agg = mean,
  minchar = 2,
  word_size = c(2, 8),
  top_words = 100,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_ggforge",
  theme_args = list(),
  palette = "Spectral",
```

```
  palcolor = NULL,
  alpha = 1,
  palreverse = FALSE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to plot
word_by	Column name to use as the word (character column).
sentence_by	Column name to split sentences into words (character column). Either word_by or sentence_by should be specified.
count_by	Column name for word/sentence counts (numeric). If NULL, counts will be computed.
score_by	Column name for word scores (numeric), used for color. If NULL, score will be set to 1.
count_name	Name for the count legend.
score_name	Name for the score legend.
split_by	Column name(s) to split data into multiple plots
split_by_sep	Separator when concatenating multiple split_by columns
words_excluded	Character vector of words to exclude from word cloud.
score_agg	Function to aggregate scores (default: mean).
minchar	Minimum number of characters for words.
word_size	Numeric vector specifying range of word sizes.
top_words	Number of top words to show.
facet_by	Column name(s) for faceting the plot
facet_scales	Scales for facets: "fixed", "free", "free_x", "free_y"
facet_ncol	Number of columns in facet layout
facet_nrow	Number of rows in facet layout
facet_byrow	Fill facets by row (TRUE) or column (FALSE)
theme	Theme name (string) or theme function
theme_args	List of arguments passed to theme function

palette	Color palette name
palcolor	Custom colors for palette
alpha	Transparency level (0-1)
palreverse	Whether to reverse the palette colors.
aspect.ratio	Aspect ratio of plot panel
legend.position	Legend position: "none", "left", "right", "bottom", "top"
legend.direction	Legend direction: "horizontal" or "vertical"
title	Plot title
subtitle	Plot subtitle
combine	Whether to combine split plots into one
nrow	Number of rows when combining plots
ncol	Number of columns when combining plots
byrow	Fill combined plots by row
seed	Random seed for reproducibility
axes	How to handle axes in combined plots ("keep", "collect", "collect_x", "collect_y")
axis_titles	How to handle axis titles in combined plots
guides	How to handle guides in combined plots ("collect", "keep", "auto")
design	Custom layout design for combined plots

Value

A ggplot object, patchwork object (if combine=TRUE), or list of plots

Examples

```
## Not run:
# Create sample data with words
data <- data.frame(
  word = rep(c(
    "apple", "banana", "cherry", "date", "elderberry",
    "fig", "grape", "kiwi", "lemon", "mango"
  ), each = 5),
  count = sample(1:50, 50, replace = TRUE),
  score = rnorm(50, mean = 5, sd = 2),
  group = rep(c("A", "B"), 25)
)

# Basic word cloud from words
WordCloudPlot(data, word_by = "word", count_by = "count", score_by = "score")

# Word cloud with custom palette
WordCloudPlot(data,
  word_by = "word", count_by = "count", score_by = "score",
  palette = "RdYlBu", palreverse = TRUE
)

# Word cloud with split by group
```

```
WordCloudPlot(data,
  word_by = "word", count_by = "count", score_by = "score",
  split_by = "group", combine = TRUE
)

# Create sample data with sentences
sentence_data <- data.frame(
  text = c(
    "Data science is an interdisciplinary field",
    "Machine learning uses statistical techniques",
    "Artificial intelligence is transforming industries",
    "Deep learning is a subset of machine learning"
  ),
  score = c(1, 2, 3, 4)
)

# Word cloud from sentences (auto-splitting)
WordCloudPlot(sentence_data,
  sentence_by = "text",
  score_by = "score",
  top_words = 20
)

## End(Not run)
```

`words_excluded` *Excluded Words for Text Analysis*

Description

Common words to exclude from keyword enrichment and text mining. Includes stop words and domain-specific common terms.

Usage

```
words_excluded
```

Format

A character vector of words to exclude

Index