Package 'plotthis'

November 6, 2026

Title High-Level Plotting Built Upon 'ggplot2' and Other Plotting Packages

Version 0.14.1

Description Provides high-level API and a wide range of options to create stunning, publication-quality plots effortlessly.
It is built upon 'ggplot2' and other plotting packages, and is designed to be easy to use and to work seamlessly with 'ggplot2' objects.
It is particularly useful for creating complex plots with multiple layers, facets, and annotations.
It also provides a set of functions to create plots for specific types of data, such as Venn diagrams, alluvial diagrams, and phylogenetic trees.
The package is designed to be flexible and customizable, and to work well with the 'ggplot2' ecosystem.
The API can be found at <https://pwwang.github.io/plotthis/reference/index.html>.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/pwwang/plotthis
https://pwwang.github.io/plotthis/

BugReports https://github.com/pwwang/plotthis/issues

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

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

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

LazyData true

Additional_repositories https://bioconductor.org/packages/release/bioc

Config/Needs/website rmarkdown

NeedsCompilation no

Author Panwen Wang [aut, cre] (ORCID: <https://orcid.org/0000-0002-4614-8970>)

Maintainer Panwen Wang <pwwang@pwwang.com>

Repository CRAN

Date/Publication 2026-11-06 14:40:02 UTC

Contents

AreaPlot . . . 3
BarPlot . . . 8
BoxPlot . . . 20
ChordPlot . . . 35
ClustreePlot . . . 39
CorPairsPlot . . . 43
CorPlot . . . 48
DensityPlot . . . 53
DimPlot . . . 61
dim_example . . . 77
DotPlot . . . 78
element_textbox . . . 87
EnrichMap . . . 89
enrich_example . . . 95
enrich_multidb_example . . . 96
GSEASummaryPlot . . . 96
gsea_example . . . 101
Heatmap . . . 101
JitterPlot . . . 119
LinePlot . . . 126
LinkedHeatmap . . . 133
ManhattanPlot . . . 142
Network . . . 150
palette_list . . . 158
palette_this . . . 162
PieChart . . . 163
QQPlot . . . 168
RadarPlot . . . 173
RarefactionPlot . . . 180
RidgePlot . . . 184
RingPlot . . . 190
ROCCurve . . . 194
SankeyPlot . . . 202
ScatterPlot . . . 210
show_palettes . . . 216
SpatImagePlot . . . 217
theme_blank . . . 230

theme_box 231
theme_this 232
TrendPlot 232
UpsetPlot 237
VelocityPlot 241
VennDiagram 246
VolcanoPlot 250
WordCloudPlot 257
words_excluded 262

Index **263**

`AreaPlot` *Area plot*

Description

Draws a stacked area plot showing how one or more groups' numeric values (or counts) accumulate across the progression of a discrete x-axis variable. Each group is rendered as a filled area stacked from baseline, making it easy to compare both individual magnitudes and the total across categories.

The function supports **count aggregation** (omit `y` to plot observation counts per x-category), **proportion scaling** (`scale_y = TRUE` normalises each x position to 100\ colour control, faceting, and splitting into separate sub-plots via `split_by`.

Usage

```
AreaPlot(
  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_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  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 = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string used to join multiple x columns. Default "_". Ignored when x is a single column.
split_by	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the fill legend title. When NULL, the group_by column name is used.
scale_y	A logical value. When TRUE, y-values are scaled to proportions within each (x, facet_by) group so that each x position stacks to 1.0. The y-axis labels switch from numeric to percent format automatically.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

x_text_angle — A numeric value specifying the angle of the x-axis text.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

seed	A numeric seed for reproducibility. Passed to validate_common_args().
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.
ncol, nrow	Integer number of columns / rows for the combined layout (passed to wrap_plots).
byrow	Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).
axes	A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides (legends) should be collected across panels. Default "collect" (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).
...	Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (x, split_by, group_by, facet_by).
2. The split_by column is validated and its NA / empty levels are processed via process_keep_na_empty(). It is then removed from the per-column keep_na / keep_empty lists.
3. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
5. AreaPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
6. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
set.seed(8525)
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)
)
# Basic stacked area
AreaPlot(data, x = "x", y = "y", group_by = "group")

# Scaled to proportions
AreaPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)

# Split into sub-plots (no group_by — single-colour fill)
AreaPlot(data, x = "x", y = "y", split_by = "group")

# Per-split palettes
AreaPlot(data, x = "x", y = "y", split_by = "group",
         palette = c(F1 = "Blues", F2 = "Reds"))

# Per-split legend positioning
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"))

# How keep_na and keep_empty work
data <- data.frame(
    x = factor(rep(c("A", NA, "C", "D"), 3),
               levels = c("A", "B", "C", "D")),
    y = c(1, 3, 6, 4, 2, 5, 7, 8, 4, 2, 3, 5),
    group = factor(sample(rep(c("F1", NA, "F3"), each = 4)),
                   levels = c("F1", "F2", "F3")),
    split = factor(sample(rep(c("X", "Y", NA), 4)),
                   levels = c("X", "Y", "Z")),
    facet = factor(sample(rep(c("M", "N", NA), 4)),
                   levels = c("M", "N", "O"))
)

# Default: NA and empty levels dropped
AreaPlot(data, x = "x", y = "y", group_by = "group")

# Keep NA and empty levels
AreaPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = TRUE, keep_empty = TRUE)

# Keep NA, assign empty levels colours but don't show them
AreaPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = TRUE, keep_empty = "level")
```

```
# Drop NA, keep empty levels
AreaPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = FALSE, keep_empty = TRUE)

# Per-column keep_na / keep_empty via named lists
AreaPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = list(x = TRUE, group = FALSE),
         keep_empty = list(x = FALSE, group = TRUE))
AreaPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = list(x = FALSE, group = TRUE),
         keep_empty = list(x = TRUE, group = FALSE))
```

`BarPlot` *Bar / SplitBar / Waterfall Plot*

Description

`BarPlot` draws bar plots with flexible fill, grouping, labelling, and annotation options. Supports both simple single-colour bars and grouped bars (dodged or stacked). Bars can be filled by a categorical variable (discrete colour scale), a continuous variable (colour gradient), or a fixed colour.

The function supports **count aggregation** (omit `y` to plot observation counts), **proportion scaling** (`scale_y = TRUE` for grouped bars), background stripes (`add_bg`), bar labels, trend lines, horizontal reference lines, and splitting into separate sub-plots via `split_by`.

`SplitBarPlot` (also known as `WaterfallPlot`) draws a divergent bar plot where bars extend left (negative values) and right (positive values) from a central zero line. The bar fill colour and opacity can encode additional variables, and the vertical ordering of categories is fully customisable.

The function supports **split_by** to produce separate panels, **facet_by** for grouped views within panels, and **alpha_by** for encoding a secondary numeric variable via opacity.

Usage

```
BarPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  flip = FALSE,
  fill_by = TRUE,
  fill_name = NULL,
  line_name = NULL,
  label_nudge = 0.02,
  label = NULL,
  label_fg = "black",
  label_size = 4,
  label_bg = "white",
  label_bg_r = 0.1,
```

```
  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,
  facet_args = list(),
  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,
  add_trend = FALSE,
  trend_color = "black",
  trend_linewidth = 1,
  trend_ptsize = 2,
  theme = "theme_this",
  theme_args = list(),
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  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,
  keep_na = FALSE,
  expand = waiver(),
```

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

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_name = "direction",
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = 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,
  keep_na = 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_name = "direction",
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
```

```
  palreverse = FALSE,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = 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,
  keep_na = 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	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to join multiple x columns. Default "_".
y	A character string specifying the column name of the data frame to plot for the y-axis.
flip	Logical; if TRUE, swap the x and y axes.
fill_by	A variable used to fill the bars. Both categorical and numeric columns are accepted: • TRUE (default) — fill by the x-axis values. • FALSE — solid fill (first palette colour). • A column name (character/factor) — discrete colour scale.

- A column name (numeric) — continuous gradient with quantile / cutoff controls.

Ignored when group_by is provided (fill is determined by group_by).

fill_name — A character string for the fill legend title. Only applies when group_by = NULL and the fill is from fill_by.

line_name — Legend name for the reference line.

label_nudge — A numeric value controlling the distance between labels and the bar top, expressed as a fraction of the data range.

label — A column name (or TRUE) for text labels on bars. When TRUE, the y-axis values are labelled. When a column name, the values in that column are used.

label_fg — A character string specifying the label text colour.

label_size — A numeric value specifying the label text size.

label_bg — A character string specifying the label background colour.

label_bg_r — A numeric value specifying the label background corner radius.

group_by — Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator

group_by_sep — The separator for multiple group_by columns. See group_by

group_name — A character string for the group fill legend title. When NULL, the group_by column name is used.

split_by — The column(s) to split the data by for separate sub-plots.

split_by_sep — Separator for concatenated split_by columns.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

facet_args — A list of additional arguments passed to the faceting function (e.g., scales, labeller).

add_bg — Logical; add alternating background stripes behind the bars.

bg_palette — Palette for the background stripes.

bg_palcolor — Custom colours for the background stripes.

bg_alpha — Alpha transparency for the background stripes.

add_line — A numeric y-intercept for a horizontal reference line.

line_color — Colour of the reference line.

line_width — Width of the reference line.

line_type — Linetype of the reference line (e.g., 1 = solid, 2 = dashed).

add_trend — Logical; add a trend line and points connecting the bar tops.

trend_color — Colour of the trend line.

trend_linewidth — Width of the trend line.

trend_ptsize — Size of the trend line points.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

lower_quantile, upper_quantile — Lower and upper quantiles for the continuous color/fill scale. The actual cutoffs are determined by these quantiles when lower_cutoff and upper_cutoff are NULL. Defaults: lower_quantile = 0, upper_quantile = 0.99.

lower_cutoff, upper_cutoff — Explicit lower and upper cutoffs for the continuous color/fill scale. When NULL (the default), the cutoffs are determined by lower_quantile and upper_quantile via quantile. Values outside the [lower_cutoff, upper_cutoff] range are clamped (winsorized) to the nearest cutoff value.

x_text_angle — A numeric value specifying the angle of the x-axis text.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

y_min, y_max — Numeric limits for the y-axis (or x-axis when flipped).

position — A character string specifying the bar layout: "auto" (default: dodge when $\leq$5 groups, stack otherwise), "dodge" (side-by-side), or "stack" (stacked on top of each other).

position_dodge_preserve — A character string passed to position_dodge2(): "total" preserves the overall bar group width; "single" preserves individual bar widths.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

expand — The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

width — A numeric value specifying the bar width (0–1).

combine — Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of ggplot objects.

ncol, nrow — Integer number of columns / rows for the combined layout.

byrow — Logical; fill the combined layout by row (default TRUE).

seed — A numeric seed for reproducibility.

axes, axis_titles — Character strings for axis handling in the combined layout.

guides — Character string for legend collection across panels.

design — A custom layout design for the combined plot.

... — Additional arguments.

y_sep	A character string to join multiple y columns. Default "_".
alpha_by	A character string naming a numeric column to encode as bar opacity. Default NULL (all bars fully opaque).
alpha_reverse	Logical; if TRUE, reverse the alpha scale direction (solid for low values, transparent for high).
alpha_name	A character string for the alpha legend title.
order_y	A named list controlling the vertical ordering of bars. Keys are "+" (positive bars), "-" (negative bars), or "*" (all bars). Values are character vectors of ordering criteria: "x_asc", "x_desc", "alpha_asc", "alpha_desc". Default orders positive bars by descending x and descending alpha; negative bars by descending x and ascending alpha.
bar_height	A numeric value (0–1) specifying the bar height as a fraction of the available category slot.
lineheight	A numeric value controlling the line height of wrapped category labels.
max_charwidth	An integer specifying the maximum character width for wrapping category labels.
fill_by_sep	A character string to join multiple fill_by columns. Default "_".
direction_name	A character string naming the internal direction column (used in legends). Default "direction".
direction_pos_name	A character string labelling the positive direction in the legend. Default "positive".
direction_neg_name	A character string labelling the negative direction in the legend. Default "negative".
x_min, x_max	Numeric limits for the x-axis. When NULL, symmetric limits are computed from the maximum absolute x-value.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (x, split_by, facet_by, group_by).
2. The split_by column is validated and its NA / empty levels are processed. It is then removed from the per-column keep_na / keep_empty lists.
3. The data is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().

5. BarPlotAtomic() is called for each split.
6. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments.
2. The split_by column is validated and its NA / empty levels are processed. It is then removed from the per-column lists.
3. The data is split by split_by (preserving level order).
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved.
5. SplitBarPlotAtomic() is called for each split. When title is a function, it receives the split level name for dynamic title generation.
6. Results are combined via combine_plots().

Examples

```
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")
)

# Single-colour bars
BarPlot(data, x = "x", y = "y")

# Solid fill (no colour mapping)
BarPlot(data, x = "x", y = "y", fill_by = FALSE)

# Label bar tops
BarPlot(data, x = "x", y = "y", label = TRUE)
BarPlot(data, x = "x", y = "y", label = "facet", label_nudge = 0)

# Grouped bars
BarPlot(data, x = "group", y = "y", group_by = "x")

# Dodged bars with background stripes
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE)

# split_by with faceting
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    facet_by = "facet", position = "dodge", facet_ncol = 1)

# split_by with collected guides
BarPlot(data,
```

```
    x = "x", y = "y", split_by = "group", facet_by = "facet",
    position = "dodge", facet_ncol = 1, guides = 'collect')

# Per-split palettes
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"),
    facet_by = "facet", position = "dodge", facet_ncol = 1)

# Background stripe palette
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE, bg_palette = "Spectral")

# Count bars (y = NULL)
BarPlot(data, x = "group", ylab = "count")

# Flipped axes
BarPlot(data, x = "group", flip = TRUE, ylab = "count")

# Numeric fill_by with colour gradient
BarPlot(data, x = "x", y = "y", fill_by = "y", flip = TRUE)

# Control fill colour scale limits (quantile)
BarPlot(data, x = "x", y = "y", fill_by = "y", flip = TRUE,
        lower_quantile = 0.1, upper_quantile = 0.9)

# Control fill colour scale limits (explicit cutoff)
BarPlot(data, x = "x", y = "y", fill_by = "y", flip = TRUE,
        lower_cutoff = 5, upper_cutoff = 12)

# keep_na and keep_empty examples
data <- data.frame(
    x = factor(c("A", "B", "C", "D", "E", "F", NA, "H"),
               levels = LETTERS[1:10]),
    y = c(10, 8, 16, 4, 6, NA, 14, 2),
    group = factor(c("G1", "G1", "G2", NA, "G3", "G3", "G5", "G5"),
                   levels = c("G1", "G2", "G3", "G4", "G5")),
    facet = factor(c("F1", NA, "F3", "F4", "F1", "F2", "F3", "F4"),
                   levels = c("F1", "F2", "F3", "F4", "F5"))
)

# Default: NA and empty levels dropped
BarPlot(data, x = "x", y = "y")

# Keep both NA and empty levels
BarPlot(data, x = "x", y = "y",
        keep_na = TRUE, keep_empty = TRUE)

# With faceting
BarPlot(data, x = "x", y = "y",
        keep_na = TRUE, keep_empty = TRUE, facet_by = "facet")
```

```
# Keep NA, hide empty levels but reserve their colours
BarPlot(data, x = "x", y = "y",
        keep_na = TRUE, keep_empty = 'level')

# Per-column keep_na / keep_empty
BarPlot(data, x = "x", y = "y",
        keep_na = list(x = TRUE), keep_empty = list(x = FALSE))

# Grouped bars with keep_na / keep_empty
BarPlot(data, x = "group", y = "y", group_by = "x")
BarPlot(data, x = "group", y = "y", group_by = "x",
        keep_na = TRUE, keep_empty = TRUE)
BarPlot(data, x = "group", y = "y", group_by = "x",
        keep_na = TRUE, keep_empty = TRUE, facet_by = "facet")

# Per-column on grouped bars
BarPlot(data, x = "group", y = "y", group_by = "x",
        keep_na = list(x = TRUE, group = FALSE),
        keep_empty = list(x = FALSE, group = TRUE))


set.seed(8525)
data <- data.frame(
    word = c("apple", "banana", "cherry", "date", "elderberry",
             "It is a very long term with a lot of words"),
    count = c(-10, 20, -30, 40, 50, 34),
    score = c(1, 2, 3, 4, 5, 3.2),
    group = c("A", "A", "B", "B", "C", "C")
)

# Basic split bar plot with alpha encoding
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score")

# Control label wrapping
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             max_charwidth = 30, lineheight = 1.1)

# Fill by categorical variable
SplitBarPlot(data, x = "count", y = "word", fill_by = "group")

# Faceting
SplitBarPlot(data, x = "count", y = "word", facet_by = "group",
             fill_name = "Direction")

# Per-split palettes
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             split_by = "group",
             palette = c(A = "Reds", B = "Blues", C = "Greens"))

# keep_na and keep_empty examples
data <- data.frame(
    word = factor(c("apple", "banana", "cherry", NA, "elderberry",
         "It is a very long term with a lot of words"),
```

```
          levels = c("apple", "banana", "cherry", "date", "elderberry",
              "unused", "It is a very long term with a lot of words")),
    count = c(-10, 20, NA, 40, 10, 34),
    score = c(1, 2, 3, 4, 5, 3.2),
    group = factor(sample(c("A", "A", "B", "B", "C", "C")),
          levels = c("A", "B", "C", "D"))
)

# Default: NA and empty levels dropped
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score")

# Keep NA and empty levels
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             keep_na = TRUE, keep_empty = TRUE)

# Keep with faceting
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             keep_na = TRUE, keep_empty = TRUE, facet_by = "group")

# Keep NA, hide empty levels (reserve colours)
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             keep_na = TRUE, keep_empty = "level")

# Per-column control
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             keep_na = list(word = FALSE), keep_empty = list(word = TRUE))

# Control fill colour scale limits
SplitBarPlot(data, x = "count", y = "word", fill_by = "score",
             lower_cutoff = 1, upper_cutoff = 4)
```

`BoxPlot` *Box / Violin / Bar / Beeswarm plot*

Description

`BoxPlot` draws box plots or bar plots (mean ± error bars) with extensive customisation options. Supports jittered or beeswarm points, paired observations with connecting lines, trend lines, statistical test annotations (pairwise or omnibus), background stripes, reference lines, point highlighting, and custom summary statistic overlays.

This is the public API — it delegates to `BoxViolinPlot` with `base = "box"` or `base = "bar"`, which in turn dispatches to `BoxViolinPlotAtomic` for each `split_by` level.

`ViolinPlot` draws violin plots with extensive customisation options. Supports jittered or beeswarm points, box plot overlays, trend lines, statistical test annotations, background stripes, reference lines, point highlighting, and custom summary statistic overlays.

This is the public API — it delegates to `BoxViolinPlot` with `base = "violin"`, which dispatches to `BoxViolinPlotAtomic` for each `split_by` level.

`BeeswarmPlot` draws beeswarm plots — points arranged by the beeswarm algorithm to avoid overlap while displaying the distribution density. This is a convenience wrapper that delegates to `BoxViolinPlot` with `base = "none"` and `add_beeswarm = TRUE`.

Requires the **ggbeeswarm** package. To get a beeswarm plot WITH a box plot, use `BeeswarmPlot(..., add_box = TRUE)`. To get a violin plot with beeswarm points, use `ViolinPlot(..., add_beeswarm = TRUE)`.

Usage

```
BoxPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  base = c("box", "bar"),
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = NULL,
  flip = FALSE,
  keep_empty = FALSE,
  keep_na = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = ifelse(isTRUE(flip), 0, 45),
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  palreverse = FALSE,
  position_dodge_preserve = "total",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  add_point = FALSE,
  pt_color = if (isTRUE(add_beeswarm)) NULL else "grey30",
  pt_size = NULL,
  pt_alpha = 1,
  jitter_width = NULL,
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
```

```
  y_min = NULL,
  y_brackets = NULL,
  add_beeswarm = FALSE,
  beeswarm_method = "swarm",
  beeswarm_cex = 1,
  beeswarm_priority = "ascending",
  beeswarm_dodge = 0.9,
  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_errorbar = "SEM",
  errorbar_color = "grey20",
  errorbar_width = 0.4,
  errorbar_linewidth = 0.6,
  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,
  ...
)

ViolinPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = NULL,
  flip = FALSE,
  keep_empty = FALSE,
  keep_na = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = ifelse(isTRUE(flip), 0, 45),
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  palreverse = FALSE,
  position_dodge_preserve = "total",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  add_point = FALSE,
  pt_color = if (isTRUE(add_beeswarm)) NULL else "grey30",
  pt_size = NULL,
  pt_alpha = 1,
  jitter_width = NULL,
```

```
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  y_brackets = NULL,
  add_beeswarm = FALSE,
  beeswarm_method = "swarm",
  beeswarm_cex = 1,
  beeswarm_priority = "ascending",
  beeswarm_dodge = 0.9,
  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,
  ...
)

BeeswarmPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = NULL,
  flip = FALSE,
  keep_empty = FALSE,
  keep_na = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = ifelse(isTRUE(flip), 0, 45),
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  palreverse = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  pt_color = NULL,
  pt_size = NULL,
  pt_alpha = 1,
```

```
  position_dodge_preserve = "total",
  jitter_width = NULL,
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  y_brackets = NULL,
  add_violin = FALSE,
  beeswarm_method = "swarm",
  beeswarm_cex = 1,
  beeswarm_priority = "ascending",
  beeswarm_dodge = 0.9,
  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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to join multiple x columns. Default "_".
y	A character string specifying the column name of the data frame to plot for the y-axis.
base	A character string: "box" (default) or "bar". Bar plots show group means with optional error bars.
in_form	A character string: "long" (default) or "wide". In wide form, x columns are pivoted to long format.
split_by	The column(s) to split the data by for separate sub-plots.
split_by_sep	Separator for concatenated split_by columns.
symnum_args	A list of arguments passed to symnum for symbolic p-value coding.
sort_x	An R expression string (e.g., "mean(y)") to order x-axis categories. Default NULL keeps the original order. When keep_empty_x is TRUE, empty levels are placed last.
flip	Logical; if TRUE, swap the x and y axes.
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot.

- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

group_by
: Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator

group_by_sep
: The separator for multiple group_by columns. See group_by

group_name
: A character string for the dodge legend title.

paired_by
: A character string naming a column that identifies paired observations. Forces add_point = TRUE and connects paired observations with lines.

x_text_angle
: A numeric value specifying the angle of the x-axis text.

step_increase
: Fractional step increase for stacking significance brackets when multiple comparisons exist.

fill_mode
: A character string controlling fill colour mapping: "dodge" (fill by group_by, discrete), "x" (fill by x-axis categories, discrete), "mean" or "median" (fill by pre-computed statistic, continuous gradient).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

position_dodge_preserve
: Passed to position_dodge(): "total" preserves the overall group width; "single" preserves individual element width.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha
: A numeric value specifying the transparency of the plot.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

`legend.direction`
: A character string specifying the direction of the legend.

`add_point`
: Logical; add jittered or beeswarm points to the plot.

`pt_color`
: Colour of the points. When `add_beeswarm = TRUE` and `pt_color` is `NULL`, points are coloured by the fill variable.

`pt_size`
: Numeric size of the points. Default computed from data size: `min(3000 / nrow(data), 0.6)`.

`pt_alpha`
: Numeric transparency of the points.

`jitter_width`
: Numeric width of the jitter. Defaults to `0.5`, but set to `0` when `paired_by` is provided.

`jitter_height`
: Numeric height of the jitter. Default `0`.

`stack`
: Logical; stack facetted panels in a compact layout with shared strip labels.

`y_max, y_min`
: Numeric y-axis limits, or quantile notation strings (e.g., `"q95"` for the 95th percentile, `"q5"` for the 5th percentile).

`y_brackets`
: Numeric y-axis position for significance brackets (or p-value labels for multiple comparisons). If NULL, the brackets are placed above the maximum y-value.

`add_beeswarm`
: Logical; use `ggbeeswarm::geom_beeswarm()` for non-overlapping point layout instead of jitter. Requires the `ggbeeswarm` package.

`beeswarm_method`
: Beeswarm layout method: `"swarm"`, `"compactswarm"`, `"hex"`, `"square"`, or `"center"`.

`beeswarm_cex`
: Numeric scaling for point spacing. Larger values spread points more.

`beeswarm_priority`
: Point layout priority: `"ascending"`, `"descending"`, `"density"`, or `"random"`.

`beeswarm_dodge`
: Numeric dodge width for beeswarm points when `group_by` is provided. Default `0.9`.

`add_trend`
: Logical; add trend lines connecting group medians.

`trend_color`
: Colour of the trend line. When `NULL` and `group_by` is present, lines are coloured per group.

`trend_linewidth`
: Width of the trend line.

`trend_ptsize`
: Size of the trend line points.

`add_stat`
: A summary function (e.g., `mean`, `median`) to display as a point with a shape legend entry.

`stat_name`
: Legend title for the stat summary shape.

`stat_color`
: Colour of the stat summary point.

`stat_size`
: Size of the stat summary point.

`stat_stroke`
: Stroke width of the stat summary point.

`stat_shape`
: Shape (an integer) for the stat summary point. Uses `scale_shape_identity()` so the shape is rendered directly.

`add_errorbar`
: Type of error bars for bar plots. See Details.

errorbar_color, errorbar_width, errorbar_linewidth
: Error bar appearance controls.

add_bg
: Logical; add alternating background stripes.

bg_palette
: Palette for the background stripes.

bg_palcolor
: Custom colours for the background stripes.

bg_alpha
: Alpha transparency for the background stripes.

add_line
: A numeric y-intercept for a horizontal reference line.

line_color
: Colour of the reference line.

line_width
: Width of the reference line.

line_type
: Linetype of the reference line.

highlight
: A specification of points to highlight: TRUE (all), a numeric index vector, a logical expression string, or a character vector of row names.

highlight_color
: Colour of highlighted points.

highlight_size
: Size of highlighted points.

highlight_alpha
: Alpha of highlighted points.

comparisons
: A logical value (TRUE for all pairs) or a list of two-element vectors specifying pairwise comparisons. Only available when fill_mode = "dodge" (i.e., group_by is present).

ref_group
: A character string specifying the reference group for comparisons.

pairwise_method
: Method for pairwise tests. Default "wilcox.test".

multiplegroup_comparisons
: Logical; perform an omnibus test (e.g., Kruskal-Wallis) across all groups.

multiple_method
: Method for the omnibus test. Default "kruskal.test".

sig_label
: Label format for significance annotations. For pairwise comparisons: "p.format", "p.signif", or a **glue** template (e.g., "p = {p}"). For multiple-group tests: "p.format" or "p.signif".

sig_labelsize
: Size of the significance label text.

hide_ns
: Logical; hide non-significant comparison labels.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
seed	A numeric seed for reproducibility.
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of ggplot objects.
ncol, nrow	Integer number of columns / rows for the combined layout.
byrow	Logical; fill the combined layout by row (default TRUE).
axes, axis_titles	Character strings for axis handling in the combined layout.
guides	Character string for legend collection across panels.
...	Additional arguments.
add_box	Logical; overlay a box plot on the primary geometry. Mutually exclusive with base = "box" and base = "bar".
box_color	Colour of the overlaid box plot outline and fill.
box_width	Width of the overlaid box plot.
box_ptsize	Size of the median point in the overlaid box plot.
add_violin	Logical; whether to add a violin plot behind the beeswarm points. **Not supported** — the function will stop with an error directing you to use ViolinPlot(..., add_beeswarm = TRUE) instead.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

Bar plots (base = "bar")

When base = "bar", bars display group means with optional error bars. add_errorbar controls the error bar type:

- "SEM" (default) — standard error of the mean.
- "SD" — standard deviation.
- "CI" or "CI95" — 95\
- "none" — no error bars.

Error bars are computed via a custom stat_summary(fun.data = ...) that handles per-group mean, SD, and sample size.

Examples

```
set.seed(8525)
data <- data.frame(
    x = rep(LETTERS[1:8], each = 40),
    y = c(rnorm(160), rnorm(160, mean = 1)),
    group1 = sample(c("g1", "g2"), 320, replace = TRUE),
    group2 = sample(c("h1", "h2", "h3", "h4"), 320, replace = TRUE)
)

# Basic box plot
BoxPlot(data, x = "x", y = "y")

# With beeswarm points
BoxPlot(data, x = "x", y = "y", add_beeswarm = TRUE, pt_color = "grey30")

# Stacked + flipped + faceted
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, facet_by = "group1",
    add_bg = TRUE, bg_palette = "Paired")

# Stacked + flipped + split_by with per-split colours
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, split_by = "group1",
    add_bg = TRUE, bg_palette = "Paired",
    palcolor = list(g1 = c("red", "blue"), g2 = c("blue", "red")))

# sort_x = order by mean(y)
data <- data.frame(
  x = factor(rep(LETTERS[1:5], each = 40),
     levels = c(LETTERS[1:2], "unused", LETTERS[3:5])),
  y = c(rnorm(40, mean = 5), rnorm(40, mean = 4), rnorm(40, mean = 3),
     rnorm(40, mean = 2), rnorm(40, mean = 1))
)
BoxPlot(data, x = "x", y = "y", sort_x = "mean(y)", keep_empty = TRUE)
BoxPlot(data, x = "x", y = "y", sort_x = "mean(-y)", keep_empty = TRUE)

# Wide-form data
data_wide <- data.frame(A = rnorm(100), B = rnorm(100), C = rnorm(100))
BoxPlot(data_wide, x = c("A", "B", "C"), in_form = "wide")

# Paired observations with connecting lines and paired test
paired_data <- data.frame(
    subject = rep(paste0("s", 1:10), each = 2),
    visit = rep(c("pre", "post"), times = 10),
    value = rnorm(20))
BoxPlot(paired_data,
    x = "visit", y = "value", comparisons = TRUE,
    paired_by = "subject", add_point = TRUE)

# Paired + grouped
```

```
paired_group_data <- data.frame(
    subject = rep(paste0("s", 1:6), each = 2),
    x = rep(c("A", "B"), each = 6),
    group = rep(c("before", "after"), times = 6),
    value = rnorm(12))
BoxPlot(paired_group_data,
    x = "x", y = "value",
    paired_by = "subject", group_by = "group",
    comparisons = TRUE, pt_size = 3, pt_color = "red")

# keep_na and keep_empty examples
data <- data.frame(
    x = factor(rep(c(LETTERS[1:3], NA, LETTERS[5:8]), each = 40),
       levels = c(LETTERS[1:8])),
    y = c(rnorm(160), rnorm(160, mean = 1)),
    group1 = sample(c("g1", "g2"), 320, replace = TRUE),
    group2 = factor(sample(c("h1", NA, "h3", "h4"), 320, replace = TRUE),
       levels = c("h1", "h2", "h3", "h4")))

BoxPlot(data, x = "x", y = "y")
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE)
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE,
        facet_by = "group2")
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = 'level')
BoxPlot(data, x = "x", y = "y", group_by = "group2")
BoxPlot(data, x = "x", y = "y", group_by = "group2",
        keep_na = TRUE, keep_empty = TRUE)
BoxPlot(data, x = "x", y = "y", group_by = "group2",
        keep_na = TRUE, keep_empty = 'level')

# Per-column keep_na / keep_empty
BoxPlot(data, x = "x", y = "y", group_by = "group2",
        keep_na = list(x = TRUE, group2 = FALSE),
        keep_empty = list(x = FALSE, group2 = TRUE))

# Bar plot (base = "bar")
data$y <- abs(data$y)
BoxPlot(data, x = "x", y = "y", base = "bar")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "SD")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "CI95")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "none")
BoxPlot(data, x = "x", y = "y", base = "bar", group_by = "group1")
BoxPlot(data, x = "x", y = "y", base = "bar", add_point = TRUE)
BoxPlot(data, x = "x", y = "y", base = "bar",
        fill_mode = "mean", palette = "Blues")


ViolinPlot(data, x = "x", y = "y")
ViolinPlot(data, x = "x", y = "y", add_beeswarm = TRUE, pt_color = "grey30")
ViolinPlot(data, x = "x", y = "y", add_box = TRUE)
ViolinPlot(data, x = "x", y = "y", add_point = TRUE)
ViolinPlot(data, x = "x", y = "y", add_trend = TRUE)
ViolinPlot(data, x = "x", y = "y", add_stat = mean)
```

```
ViolinPlot(data, x = "x", y = "y", add_bg = TRUE)
ViolinPlot(data, x = "x", y = "y", add_line = 0)

# Grouped
ViolinPlot(data, x = "x", y = "y", group_by = "group1")

# Grouped + faceted + box overlay
ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    facet_by = "group2", add_box = TRUE)

# Highlight
ViolinPlot(data,
    x = "x", y = "y", add_point = TRUE,
    highlight = 'group1 == "g1"', alpha = 0.8,
    highlight_size = 1.5, pt_size = 1, add_box = TRUE)

# Pairwise comparisons with formatted labels
if (requireNamespace("ggpubr", quietly = TRUE)) {
  # https://github.com/kassambara/ggpubr/issues/751
  library(ggpubr)
  ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    comparisons = TRUE, sig_label = "p = {p}")
}

# Explicit comparison list + hide non-significant
ViolinPlot(data,
    x = "x", y = "y", sig_label = "p.format", hide_ns = TRUE,
    facet_by = "group2", comparisons = list(c("D", "E")))

# Continuous fill (mean) + omnibus test
ViolinPlot(data,
    x = "x", y = "y", fill_mode = "mean",
    facet_by = "group2", palette = "Blues",
    multiplegroup_comparisons = TRUE)

# Per-split palettes
ViolinPlot(data,
    x = "x", y = "y", fill_mode = "mean",
    split_by = "group1", palette = c(g1 = "Blues", g2 = "Reds"))

# Stacked faceting
ViolinPlot(data,
    x = "x", y = "y", stack = TRUE,
    facet_by = "group2", add_box = TRUE, add_bg = TRUE,
    bg_palette = "Paired")


# Basic beeswarm
BeeswarmPlot(data, x = "x", y = "y")

# Control point size
```

```
BeeswarmPlot(data, x = "x", y = "y", pt_size = 1)

# Beeswarm with box overlay
BeeswarmPlot(data, x = "x", y = "y", add_box = TRUE, pt_color = "grey30")

# Grouped
BeeswarmPlot(data, x = "x", y = "y", group_by = "group1")

# Grouped without dodging
BeeswarmPlot(data, x = "x", y = "y", group_by = "group1",
             beeswarm_dodge = NULL)

# Hex layout with wider spacing
BeeswarmPlot(data,
    x = "x", y = "y", beeswarm_method = "hex",
    beeswarm_cex = 2)
```

`ChordPlot` *Chord / Circos plot*

Description

Draws a chord diagram (also known as a circos plot) to visualise relationships between two categorical variables. Categories are arranged around a circle, and connecting ribbons (links) represent the flow or association between source and target nodes. The width of each link is proportional to the associated numeric value or observation count.

The function supports **count aggregation** (omit `y` to plot observation counts per pair), **link colouring** by source or target node, **label rotation** options, and splitting into separate sub-diagrams via `split_by`.

`CircosPlot` is an alias of `ChordPlot`.

`CircosPlot` is an alias for `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_this",
```

```
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  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_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
```

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

Arguments

data	A data frame.
y	A character string specifying the column name of the data frame to plot for the y-axis.
from	A character string (or vector) specifying the column name(s) for the source nodes. Character/factor columns are expected. Multiple columns are concatenated with from_sep.
from_sep	A character string to join multiple from columns. Default "_".
to	A character string (or vector) specifying the column name(s) for the target nodes. Character/factor columns are expected. Multiple columns are concatenated with to_sep.
to_sep	A character string to join multiple to columns. Default "_".
split_by	The column(s) to split the data by for separate sub-diagrams. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
flip	Logical; if TRUE, swap the source and target nodes, reversing the link direction.
links_color	A character string controlling which node's colour each link ribbon takes: "from" (default) or "to".
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
labels_rot	Logical; if TRUE, rotate sector labels by 90 degrees (clockwise). Default FALSE uses niceFacing for automatic orientation.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle	A character string specifying the subtitle of the plot.
seed	A numeric seed for reproducibility.
keep_na	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot. • "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual wrapped elements.
ncol, nrow	Integer number of columns / rows for the combined layout.
byrow	Logical; fill the combined layout by row (default TRUE).
axes, axis_titles	Character strings for axis handling in the combined layout.
guides	Character string for legend collection across panels.
design	A custom layout design for the combined plot.
...	Additional arguments.

Value

A patchwork object or a named list of wrapped elements (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (split_by, from, to).
2. The split_by column is validated and its NA / empty levels are processed. It is then removed from the per-column lists.
3. The data is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".

4. Per-split `palette` and `palcolor` are resolved via `check_palette()` and `check_palcolor()`.
5. `ChordPlotAtomic()` is called for each split. When `title` is a function, it receives the split level name for dynamic titles.
6. Results are combined via `combine_plots()`.

Examples

```
set.seed(8525)
data <- data.frame(
    nodes1 = sample(c("Soure1", "Source2", "Source3"), 10, replace = TRUE),
    nodes2 = sample(letters[1:3], 10, replace = TRUE),
    y = sample(1:5, 10, replace = TRUE)
)

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

# Links coloured by target + rotated labels
ChordPlot(data, from = "nodes1", to = "nodes2",
          links_color = "to", labels_rot = TRUE)

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

# Split by a column — one diagram per split level
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y")

# Per-split palettes
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y",
          palette = c("1" = "Reds", "2" = "Blues",
                      "3" = "Greens", "4" = "Purp"))

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

`ClustreePlot` *Clustree Plot*

Description

Creates a clustree (clustering tree) plot visualising how cluster assignments change across increasing clustering resolutions. The plot helps identify stable clustering solutions and understand the hierarchical relationships among clusters at different resolution thresholds.

The function expects a data frame with columns named by a common `prefix` followed by numeric resolution values (e.g. `"res_0.1"`, `"res_0.3"`, `"res_0.5"`). Each column contains cluster labels (factor or character) for every observation at that resolution.

Internally, the function uses `clustree::clustree()` to compute a ggraph-based tree layout where nodes are clusters and edges represent cells transitioning between clusters at adjacent resolutions. Edge colour and width encode the number of transitioning cells.

Key features:

- **Resolution-level node colouring** — each resolution receives a distinct colour from the selected `palette`.
- **Edge gradient** — edges are coloured by transition count using a separate `edge_palette` colour gradient.
- **Flip support** — `flip = TRUE` places resolutions on the x-axis for left-to-right reading.
- **Split by groups** — `split_by` generates per-group clustree plots that are combined via `patchwork`.
- **Automatic dimensions** — plot height and width are automatically computed based on the number of resolutions, clusters, and the legend configuration.

Usage

```
ClustreePlot(
  data,
  prefix,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  edge_palette = "Spectral",
  edge_palcolor = NULL,
  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_this",
  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.
prefix	A character string specifying the common prefix of the resolution columns in data. All columns whose names start with this prefix are selected as resolution columns. The suffix after the prefix is parsed as a numeric resolution value. Supports "_" and "." as separators between the prefix and the resolution value (e.g. "res_0.5" or "p.0.5").
flip	A logical value. If TRUE, the tree is flipped so that resolutions are displayed on the x-axis (left to right) and cluster assignments are shown as row labels on the y-axis. Default: FALSE.
split_by	The column(s) to split data by and generate separate clustree plots for each level. Each split level produces an independent clustree plot via ClustreePlotAtomic.
split_by_sep	A character string used to concatenate multiple split_by column values when split_by specifies more than one column. Default: "_".
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
edge_palette	A character string specifying the palette name for the edge colour gradient. Edges are coloured by the number of transitioning cells between clusters at adjacent resolutions, using ggraph::scale_edge_color_gradientn(). Default: "Spectral".
edge_palcolor	A character vector of custom colours for the edge colour gradient. When NULL (the default), colours are derived from edge_palette.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
expand	The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is

used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`theme` A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

`theme_args` A list of arguments to pass to the theme function.

`combine` A logical value. If `TRUE` (the default), the list of per-split plots is combined into a single `patchwork` object. If `FALSE`, returns the raw list of `ggplot` objects.

`nrow`, `ncol`, `byrow`
Integers controlling the layout of combined plots via `patchwork::wrap_plots()`. `byrow = TRUE` (default) fills the layout row-wise. Ignored when `design` is provided.

`seed` The random seed for reproducibility. Passed to `validate_common_args()`. Default: `8525`.

`axes`, `axis_titles`
Strings controlling how axes and axis titles are handled across combined plots. Passed to `combine_plots()`. See `?patchwork::wrap_plots` for options (`"keep"`, `"collect"`, `"collect_x"`, `"collect_y"`).

`guides` A string controlling guide collection across combined plots. Passed to `combine_plots()`.

`design` A custom layout specification for combined plots. Passed to `combine_plots()`. When specified, `nrow`, `ncol`, and `byrow` are ignored.

`...` Additional arguments passed to `clustree::clustree()`. Commonly used overrides include `node_size_range`, `node_text_size`, `layout` (default: `"sugiyama"`), `show_axis`, and `node_text_colour`. Note that `x` (the data) and `prefix` are set internally and cannot be overridden here.

Value

A `ggplot` object (single plot), a `patchwork` object (when `combine = TRUE` with `split_by`), or a `list` of `ggplot` objects (when `combine = FALSE`).

split_by Workflow (ClustreePlot)

When `split_by` is provided, the following pipeline executes:

1. **Argument validation** — `validate_common_args()` checks the `seed` value and sets the random seed.
2. **Theme resolution** — `process_theme()` resolves the `theme` string or function to a theme function.
3. **Split column validation** — `check_columns()` resolves `split_by` with `force_factor = TRUE`, `allow_multi = TRUE`, `concat_multi = TRUE`.

4. **Data splitting** — splits `data` by `split_by` levels (unused levels dropped), preserving factor level order.
5. **Per-split palette / colour / legend** — `check_palette()`, `check_palcolor()`, and `check_legend()` resolve per-split overrides for `palette`, `palcolor`, `legend.position`, and `legend.direction`.
6. **Per-split title** — when `title` is a function, it receives the default title (the split level name) and can return a custom string; otherwise `title %||% split_level` is used.
7. **Dispatch** — each split subset is passed to `ClustreePlotAtomic` with the per-split parameters.
8. **Combination** — `combine_plots()` assembles the list of plots via `patchwork::wrap_plots`, honouring `nrow`/`ncol`/`byrow`/`design`.
9. **Combined data** — when `combine = TRUE`, the combined plot's `data` (`p$data`) exposes the node layout of every split, each row tagged with the `split_by` level, plus a combined graph of all splits' nodes and links (`"graph"` attribute) and the original links (`"edges"` attribute).

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)
)

# --- Basic clustree plot ---
ClustreePlot(data, prefix = "p")

# --- Flipped layout (resolutions on x-axis) ---
ClustreePlot(data, prefix = "p", flip = TRUE)

# --- Split by group ---
ClustreePlot(data, prefix = "p", split_by = "split")

# --- Split by group with per-split palettes ---
ClustreePlot(data, prefix = "p", split_by = "split",
             palette = c("1" = "Set1", "2" = "Paired"))
```

`CorPairsPlot` *Correlation pairs (scatterplot matrix)*

Description

Draws a grid of pairwise scatter plots for selected numeric columns, arranged in a scatterplot matrix layout. The upper or lower triangle displays correlation tiles while the opposite triangle shows scatter plots with regression lines. Diagonal cells can show density plots, violin plots, histograms, box plots, or a simple diagonal line.

NOTE: The `facet_by` parameter is **not supported** in CorPairsPlot (an error is raised if provided). Use `split_by` instead to create separate correlation pair matrices per group.

The function supports **four layout orientations** (`layout`), **three correlation methods**, configurable diagonal plots via other plotthis functions, custom correlation tile formatting, and splitting into separate sub-plots via `split_by`.

Usage

```
CorPairsPlot(
  data,
  columns = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  diag_type = NULL,
  diag_args = list(),
  layout = c(".\\", "\\.", "/.", "./"),
  cor_method = c("pearson", "spearman", "kendall"),
  cor_palette = "RdBu",
  cor_palcolor = NULL,
  cor_size = 3,
  cor_format = "corr: {round(corr, 2)}",
  cor_fg = "black",
  cor_bg = "white",
  cor_bg_r = 0.1,
  theme = "theme_this",
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  facet_by = NULL,
  legend.position = "right",
  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.
columns	A character vector of column names to include in the pairs plot. When NULL (default), all columns except group_by are used. At least two columns are required.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the colour legend title in the scatter plots. When NULL, the group_by column name is used.
split_by	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
diag_type	A character string specifying the plot type for diagonal cells. One of "density", "violin", "histogram", "box", or "none" (diagonal line). Default: "density" (no group_by) or "violin" (with group_by).
diag_args	A named list of additional arguments passed to the diagonal plot function (DensityPlot, ViolinPlot, Histogram, or BoxPlot). Default: list().
layout	A character string specifying the layout orientation. One of the following codes (dot = scatter, backslash/slash = diagonal): .\, \\., /., ./. Default: .\.
cor_method	A character string specifying the correlation method for the fill tiles. One of "pearson", "spearman", "kendall". Default: "pearson".
cor_palette	A character string specifying the colour palette for the correlation fill tiles. Default: "RdBu".
cor_palcolor	A character vector of custom colours used to create the correlation tile palette. When NULL, the palette's default colours are used.
cor_size	A numeric value specifying the font size of the correlation text in the fill tiles. Default: 3.
cor_format	A character string specifying a glue template for formatting the correlation text. The template is evaluated by glue::glue() with access to corr (the correlation value), x, and y (the column names). Default: "corr: \{round(corr, 2)\}".
cor_fg	A character string specifying the colour of the correlation text. Default: "black".
cor_bg	A character string specifying the background colour of the correlation text boxes. Default: "white".
cor_bg_r	A numeric value specifying the corner radius of the correlation text background boxes. Default: 0.1.

theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
seed	A numeric seed for reproducibility.
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual patchwork objects.
ncol, nrow	Integer number of columns / rows for the combined layout.
byrow	Logical; fill the combined layout by row. Default TRUE.
axes	A character string specifying how axes should be treated across the combined layout.
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides (legends) should be collected across panels.
design	A custom layout design for the combined plot.
...	Additional arguments.

Value

A patchwork object (when combine = TRUE) or a named list of patchwork objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. The split_by column is validated via check_columns() with force_factor = TRUE. Empty levels are dropped.
2. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...". The split_by column is removed from each split's data before plotting.
3. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
4. CorPairsPlotAtomic() is called for each split. When title is a function, it receives the split level name and can generate dynamic titles.
5. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
set.seed(8525)
data <- data.frame(x = rnorm(100))
data$y <- rnorm(100, 10, sd = 0.5)
data$z <- -data$x + data$y + rnorm(100, 20, 1)
data$g <- sample(1:4, 100, replace = TRUE)

# Histogram diagonal, slash layout
CorPairsPlot(data, diag_type = "histogram",
    diag_args = list(bins = 30, palette = "Paired"),
    layout = "/.")

# No diagonal with axis title styling
CorPairsPlot(data, group_by = "g", diag_type = "none", layout = "./",
    theme_args = list(axis.title = element_textbox(
        color = "black", box.color = "grey20", size = 16, halign = 0.5,
        fill = "grey90", linetype = 1,
        width = grid::unit(1, "npc"),
        padding = ggplot2::margin(5, 5, 5, 5))))

# Violin diagonal with custom format
CorPairsPlot(data, group_by = "g", diag_type = "violin", layout = "\\.",
    cor_format = "{x}\n{y}\ncorr: {round(corr, 2)}")

# Per-split with bottom legend
CorPairsPlot(data, split_by = "g", diag_type = "none", layout = ".\\",
    legend.position = "bottom", legend.direction = "horizontal",
    group_name = "group")

# Per-split with custom palette colours
CorPairsPlot(data, split_by = "g",
    palcolor = list("1" = "red", "2" = "blue", "3" = "green",
                    "4" = "yellow"))
```

CorPlot *Correlation scatter plot*

Description

Draws a scatter plot of two numeric variables with a linear regression line, optional correlation statistics, and point highlighting. This is the public entry point that wraps `CorPlotAtomic` with `split_by` support.

Key features include **group-based colouring** (`group_by`), **point highlighting** by expression, row-name, or index, **annotation items** (regression equation, R-squared, p-value, Spearman/Pearson/Kendall rho, N), **raster rendering** for large datasets, **faceting** (`facet_by`), and **splitting** into separate sub-plots via `split_by`.

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("eq", "r2", "p"),
  anno_size = 3,
  anno_fg = "black",
  anno_bg = "white",
  anno_bg_r = 0.1,
 anno_position = c("topleft", "topright", "bottomleft", "bottomright", "tl", "tr", "bl",
    "br"),
  add_smooth = TRUE,
  smooth_color = "red2",
  smooth_width = 1.5,
  smooth_se = FALSE,
  theme = "theme_this",
```

```
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = 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",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the colour legend title. When NULL, the group_by column name is used.
split_by	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
pt_size	A numeric value specifying the size of the points. Default: 2.
pt_shape	A numeric value specifying the shape of the points (see geom_point). Default: 16 (filled circle).

raster — A logical value. When TRUE, uses scattermore::geom_scattermore() for efficient rendering of large datasets. Default: FALSE.

alpha — A numeric value specifying the transparency of the plot.

raster_dpi — An integer vector of length 1 or 2 specifying the raster resolution in (width, height) pixels. When a single value is provided, it is recycled. Default: c(512, 512).

highlight — Specifies which points to emphasise. Can be:

- TRUE — highlight all points.
- A character expression (e.g. 'Species == "setosa"') — evaluated via dplyr::filter.
- A character vector — matched against rownames of the data.
- A numeric vector — treated as row indices.

Default: NULL (no highlighting).

highlight_color — A character string specifying the colour of the highlighted point borders. Default: "black".

highlight_size — A numeric value specifying the size of the highlighted points (the inner fill). Default: 1.

highlight_alpha — A numeric value specifying the alpha transparency of the highlighted points. Default: 1.

highlight_stroke — A numeric value specifying the stroke width of the highlighted point borders. The outer layer size is highlight_size + highlight_stroke. Default: 0.8.

anno_items — A character vector specifying which statistics to display as text annotation. Available items: "eq" (regression equation), "r2" (R-squared), "p" (p-value), "spearman", "pearson", "kendall", "n" (observation count). Default: c("eq", "r2", "p").

anno_size — A numeric value specifying the font size of the annotation text (scaled by base_size / 12). Default: 3.

anno_fg — A character string specifying the colour of the annotation text. Default: "black".

anno_bg — A character string specifying the background colour of the annotation text boxes. Default: "white".

anno_bg_r — A numeric value specifying the corner radius of the annotation text background boxes. Default: 0.1.

anno_position — A character string specifying the corner position of the annotation text. One of "topleft" (alias "tl"), "topright" ("tr"), "bottomleft" ("bl"), "bottomright" ("br").

add_smooth — A logical value. When TRUE (default), a linear regression line (geom_smooth(method = "lm")) is added.

smooth_color — A character string specifying the colour of the regression line. Default: "red2".

smooth_width — A numeric value specifying the linewidth of the regression line. Default: 1.5.

smooth_se — A logical value. When TRUE, a standard error band is drawn around the regression line. Default: FALSE.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

seed — A numeric seed for reproducibility. Passed to validate_common_args().

combine — Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow — Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow — Logical; fill the combined layout by row. Default TRUE.

axes — A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).

axis_titles — A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.

guides	A character string specifying how guides (legends) should be collected across panels (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).
...	Additional arguments.

Value

A ggplot object (when split_by is NULL), a patchwork object (when combine = TRUE), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. The split_by column is validated via check_columns() with force_factor = TRUE. Empty levels are dropped (droplevels()).
2. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
3. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
4. CorPlotAtomic() is called for each split. When title is a function, it receives the split level name and can generate dynamic titles.
5. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
data(iris)

# Basic scatter with group colours
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species")

# Highlight a specific group with custom stroke
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species",
    highlight = 'Species == "setosa"', highlight_stroke = 1.5,
    anno_items = c("eq", "pearson"), anno_position = "bottomright")

# Faceted by species
CorPlot(iris, "Sepal.Length", "Sepal.Width", facet_by = "Species",
    facet_scales = "free")

# Per-split palettes
CorPlot(iris, "Sepal.Length", "Sepal.Width", split_by = "Species",
    palette = c(setosa = "Set1", versicolor = "Dark2", virginica = "Paired"))
```

DensityPlot *Density Plot / Histogram*

Description

Density plot for visualising the distribution of a numeric variable. Uses ggplot2::geom_density() to render smooth kernel density estimates, with optional grouping, faceting, split-by splitting, and data-distribution rug bars along the baseline.

This is the public entry point for density plots; the companion Histogram() function provides binned-histogram rendering. Both dispatch to the same internal engine (DensityHistoPlotAtomic) with type = "density" or type = "histogram" respectively.

Histogram for visualising the distribution of a numeric variable via binned counts. Uses ggplot2::geom_histogram(), with optional trend-line overlays, zero-skip interpolation, grouping, faceting, and split-by splitting.

This is the histogram companion to DensityPlot(). Both dispatch to the same internal engine (DensityHistoPlotAtomic) with type = "histogram" or type = "density" respectively.

When use_trend = TRUE, the histogram bars are replaced entirely by a point-and-line trend; when add_trend = TRUE, the trend is overlaid on top of the bars. The trend_skip_zero option uses zoo::na.approx() to interpolate across empty bins for a continuous trend curve — particularly useful with transformed y-axes.

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,
  palreverse = FALSE,
  alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  keep_na = FALSE,
```

```
  keep_empty = FALSE,
  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,
  aspect.ratio = 1,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

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",
  keep_na = FALSE,
  keep_empty = FALSE,
  use_trend = FALSE,
```

```
  add_trend = FALSE,
  trend_alpha = 1,
  trend_linewidth = 0.8,
  trend_pt_size = 1.5,
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  theme = "theme_this",
  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,
  aspect.ratio = 1,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the legend title for the group_by aesthetic. When NULL (default), the (possibly concatenated) group_by column name is used.
xtrans	A character string specifying the transformation applied to the x-axis. Passed

to ggplot2::scale_x_continuous(transform = ...). Supported values include "identity" (default), "log10", "log2", "sqrt", "reverse", etc.

ytrans — A character string specifying the transformation applied to the y-axis. Passed to ggplot2::scale_y_continuous(transform = ...). Used by trend_skip_zero to correctly interpolate across zero bins on a transformed scale. Default: "identity".

split_by — The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.

split_by_sep — The separator for multiple split_by columns. See split_by

flip — A logical value. If TRUE, the x and y axes are swapped via coord_flip(). Dimension calculation accounts for the flip.

position — A character string specifying the position adjustment for the bars or density curves. Default: "identity", which shows the actual count / density per group (unlike ggplot2's default "stack"). Other options: "stack", "dodge", "fill".

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

add_bars — A logical value. If TRUE, a data-distribution rug is drawn along the y = 0 axis using geom_linerange(). Each group's bars are vertically offset to avoid overlap.

bar_height — A numeric value specifying the height (in data units, relative to the maximum y) of the rug bars added by add_bars. The actual pixel height scales with max_y. Default: 0.025.

bar_alpha — A numeric value in [0, 1] for the transparency of the rug bars. Default: 1.

bar_width — A numeric value passed as the linewidth aesthetic of geom_linerange(). Controls the thickness of each rug tick. Default: 0.1.

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the

behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

expand
: The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.
  - 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design
: Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

...
: Additional arguments.

bins
: A numeric value specifying the number of bins for the histogram. Ignored when type = "density". Defaults to 30 when neither bins nor binwidth is provided.

binwidth
: A numeric value specifying the width of individual bins for the histogram. Ignored when type = "density". Takes precedence over bins when both are set.

trend_skip_zero
: A logical value. If TRUE, bins with zero count are set to NA before the trend line is computed, and zoo::na.approx() is used to interpolate across the gaps — producing a continuous curve even when some bins are empty. Requires ytrans to be correctly specified. Only applies when type = "histogram" and use_trend or add_trend is active.

use_trend A logical value. If TRUE, the histogram bars are replaced entirely by a trend line (points + connecting line). Only applies when type = "histogram".

add_trend A logical value. If TRUE, a trend line is overlaid on top of the histogram bars. Only applies when type = "histogram".

trend_alpha A numeric value in [0, 1] controlling the transparency of the trend points and line. Default: 1.

trend_linewidth A numeric value for the thickness of the trend line. Default: 0.8.

trend_pt_size A numeric value for the size of the trend points. Default: 1.5.

Value

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE).

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE).

split_by Workflow

When split_by is specified, DensityPlot() executes the following pipeline:

1. **Argument validation** — validate_common_args() checks the seed and facet-by consistency.
2. **NA / empty normalisation** — check_keep_na() / check_keep_empty() convert keep_na / keep_empty to per-column lists.
3. **Theme resolution** — process_theme() resolves the theme string to a theme function.
4. **Split column resolution** — check_columns() validates split_by (force_factor, concat_multi).
5. **Pre-filtering** — process_keep_na_empty() removes NA / empty levels from the split column, then data is split by split_by levels (order preserved).
6. **Per-split parameter resolution** — check_palette(), check_palcolor(), check_legend() resolve palette, palcolor, legend.position, and legend.direction for each split.
7. **Per-split dispatch** — each split is passed to DensityHistoPlotAtomic(type = "density", ...) with its resolved parameters. Title defaults to the split level name unless title is a function.
8. **Combination** — combine_plots() assembles the list of plots via patchwork::wrap_plots(), applying nrow, ncol, byrow, axes, axis_titles, guides, and design.

When split_by is specified, Histogram() executes the following pipeline:

1. **Argument validation** — validate_common_args() checks the seed and facet-by consistency.
2. **NA / empty normalisation** — check_keep_na() / check_keep_empty() convert keep_na / keep_empty to per-column lists.
3. **Theme resolution** — process_theme() resolves the theme string to a theme function.
4. **Split column resolution** — check_columns() validates split_by (force_factor, concat_multi).

5. **Pre-filtering** — process_keep_na_empty() removes NA / empty levels from the split column, then data is split by split_by levels (order preserved).
6. **Per-split parameter resolution** — check_palette(), check_palcolor(), check_legend() resolve palette, palcolor, legend.position, and legend.direction for each split.
7. **Per-split dispatch** — each split is passed to DensityHistoPlotAtomic(type = "histogram", ...) with its resolved parameters (including bins, binwidth, use_trend, add_trend, trend_skip_zero, trend_alpha, trend_linewidth, trend_pt_size). Title defaults to the split level name unless title is a function.
8. **Combination** — combine_plots() assembles the list of plots via patchwork::wrap_plots(), applying nrow, ncol, byrow, axes, axis_titles, guides, and design.

Examples

```
set.seed(8525)
data <- data.frame(
    x = c(rnorm(500, -1), rnorm(500, 1)),
    group = factor(rep(c("A", NA, "C", "D"), each = 250), levels = LETTERS[1:4]),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

# basic density
DensityPlot(data, x = "x")
DensityPlot(data, x = "x", group_by = "group")

# NA / empty level handling
DensityPlot(data, x = "x", group_by = "group",
    keep_na = TRUE, keep_empty = TRUE)
DensityPlot(data, x = "x", group_by = "group",
    keep_na = TRUE, keep_empty = 'level')

# faceting and splitting
DensityPlot(data, x = "x", group_by = "group", facet_by = "facet")
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE)
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE,
    palette = c(F1 = "Set1", F2 = "Set2"))

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", NA, "D"), each = 250), levels = LETTERS[1:4]),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

# basic histogram
Histogram(data, x = "x")
Histogram(data, x = "x", group_by = "group")

# NA / empty level handling
Histogram(data, x = "x", group_by = "group", keep_na = TRUE, keep_empty = 'level')

# add_bars and trend overlays
```

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

# use_trend replaces bars entirely
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 use_trend = TRUE, trend_pt_size = 3)

# per-split palettes
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 palette = c(F1 = "Paired", F2 = "Spectral"))
```

DimPlot *DimPlot / FeatureDimPlot*

Description

DimPlot visualizes dimension reduction data (PCA, t-SNE, UMAP, etc.) as a 2D or 3D scatter plot. DimPlot() colours points by a discrete grouping variable (e.g., clusters), while FeatureDimPlot() colours points by a continuous numeric feature (e.g., gene expression, lineage scores).

Both functions share the same internal engine (DimPlotAtomic) and support an extensive set of annotation layers: group boundary marks, network/graph edges, 2D density contours, lineage/trajectory curves, RNA-velocity arrows (raw, grid, or stream), statistical summary mini-plots at group centroids, point highlighting, background context points from other facets, and flexible label positioning.

When dims has 3 elements, both functions automatically return an interactive **plotly** 3D scatter plot (via DimPlotAtomic3D). Certain 2D-only features are silently ignored in 3D mode (see @param dims for the full list).

Rendering scales with dataset size: standard geom_point() for small data, automatic rasterisation via scattermore::geom_scattermore() when nrow(data) > 1e5, or hex-bin aggregation (geom_hex() / stat_summary_hex()).

FeatureDimPlot to visualize feature expression on dimension reduction plots. Colours points by a continuous numeric variable (e.g., gene expression, module score, lineage pseudotime) using a gradient colour scale, with optional quantile trimming and background cutoff.

When multiple features are provided and facet_by is not set, the data is automatically pivoted to long format and faceted by feature name. split_by = TRUE dispatches each feature to a separate plot for independent layout control. split_by as a column name splits by that column's levels, producing one plot per level with per-split palette support.

For detailed split_by workflows, see the main DimPlot / FeatureDimPlot documentation (@section split_by Workflow (

Usage

```
DimPlot(
  data,
  dims = 1:2,
  group_by,
```

```
group_by_sep = "_",
split_by = NULL,
split_by_sep = "_",
pt_size = NULL,
pt_alpha = 1,
bg_color = "grey80",
label_insitu = FALSE,
show_stat = !identical(theme, "theme_blank"),
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",
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 = unit(3, "mm"),
mark_alpha = 0.1,
mark_linetype = 1,
stat_by = NULL,
stat_plot_type = c("pie", "ring", "bar", "line"),
stat_plot_size = 0.1,
stat_args = list(palette = "Set1"),
graph = NULL,
edge_size = c(0.05, 0.5),
edge_alpha = 0.1,
edge_color = "grey40",
add_density = FALSE,
density_color = "grey80",
density_filled = FALSE,
density_filled_palette = "Greys",
density_filled_palcolor = NULL,
lineages = NULL,
lineages_trim = c(0.01, 0.99),
lineages_span = 0.75,
lineages_palette = "Dark2",
lineages_palcolor = NULL,
lineages_arrow = arrow(length = unit(0.1, "inches")),
```

```
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  velocity = NULL,
  velocity_plot_type = c("raw", "grid", "stream"),
  velocity_n_neighbors = NULL,
  velocity_density = 1,
  velocity_smooth = 0.5,
  velocity_scale = 1,
  velocity_min_mass = 1,
  velocity_cutoff_perc = 5,
  velocity_group_palette = "Set2",
  velocity_group_palcolor = NULL,
  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,
  keep_na = FALSE,
  keep_empty = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
  raster_dpi = c(512, 512),
```

```
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = TRUE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

FeatureDimPlot(
  data,
  dims = 1:2,
  features,
  split_by = NULL,
  split_by_sep = "_",
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  bg_cutoff = NULL,
  label_insitu = FALSE,
  show_stat = !identical(theme, "theme_blank"),
  color_name = "",
  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",
  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 = unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  stat_by = NULL,
  stat_plot_type = c("pie", "ring", "bar", "line"),
  stat_plot_size = 0.1,
  stat_args = list(palette = "Set1"),
  graph = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  lineages = NULL,
  lineages_trim = c(0.01, 0.99),
  lineages_span = 0.75,
  lineages_palette = "Dark2",
  lineages_palcolor = NULL,
  lineages_arrow = arrow(length = unit(0.1, "inches")),
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  velocity = NULL,
  velocity_plot_type = c("raw", "grid", "stream"),
  velocity_n_neighbors = NULL,
  velocity_density = 1,
  velocity_smooth = 0.5,
  velocity_scale = 1,
  velocity_min_mass = 1,
  velocity_cutoff_perc = 5,
  velocity_group_palette = "Set2",
  velocity_group_palcolor = NULL,
  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,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
dims	A character vector of the column names to plot on the x, y (and optionally z) axes or a numeric vector of the column indices. When 3 dimensions are provided, a 3D interactive plot is created using plotly. Supported in 3D: group_by, features, labels, highlight, lineages, graph/network, show_stat, order. Not supported in 3D: add_mark, stat_by, add_density, velocity, hex, facet_by, raster.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
split_by	A character vector of column names to split the data by and plot separately. If TRUE, the data is split by features — each feature is plotted in its own panel. Use this instead of facet_by when you need independent layout control (nrow, ncol) or per-feature palettes via split_by = TRUE combined with palette / palcolor.
split_by_sep	The separator for multiple split_by columns. See split_by
pt_size	A numeric value for point size, or a character string naming a numeric column in data to map point sizes to. If NULL (default), auto-calculated as min(3000 / nrow(data), 0.6). Column-name mapping only applies to 2D plots and is ignored with a message when raster = TRUE.
pt_alpha	A numeric value in [0, 1] for the point transparency. Default is 1.
bg_color	A character string specifying the colour used for NA-valued points and background context points drawn from other facets. Default is "grey80".
label_insitu	A logical value. If TRUE, the raw group names are placed at the group median coordinates instead of numeric indices. Forces label = TRUE. Default is FALSE.
show_stat	A logical value. If TRUE (default), the number of points per group is shown in the legend labels and subtitle. Ignored when theme = "theme_blank".
label	A logical value. If TRUE, group labels (numeric indices by default, or group names when label_insitu = TRUE) are placed at the median coordinates of each group. Forced to TRUE when label_repel or label_insitu is set.
label_size	A numeric value for the label text size. Passed to ggrepel::geom_text_repel(). Default is 4.
label_fg	A character string for the label text (foreground) colour. Default is "white".
label_bg	A character string for the label background / outline colour. Default is "black".
label_bg_r	A numeric value for the background fill ratio of the label bounding box. Passed to ggrepel::geom_text_repel(bg.r = ...). Default is 0.1.
label_repel	A logical value. If TRUE, labels are repelled from each other with force label_repulsion. A visible point anchor is drawn. Forces label = TRUE.
label_repulsion	A numeric value for the repulsion force when label_repel = TRUE. Passed to ggrepel::geom_text_repel(force = ...). Default is 20.

label_pt_size A numeric value for the size of the anchor point drawn when label_repel = TRUE. Default is 1.

label_pt_color A character string for the colour of the label anchor point. Default is "black".

label_segment_color
A character string for the colour of the line segment connecting the label to the anchor. Used in non-repel mode (label_repel = FALSE) where min.segment.length = 0. Default is "black".

order A character string controlling the draw order of points:

- "as-is" (default) — the row order in the data is preserved.
- "reverse" — rows are reversed.
- "high-top" — points with high values (last factor levels for group_by) are drawn last (on top).
- "low-top" — points with low values (first factor levels) are drawn last.
- "random" — rows are randomly shuffled.

For high-top and low-top, NA values are always plotted at the bottom. When applied to group_by, only the draw order changes — legend colours and order are unaffected. Within the same level, point order is preserved. For precise control, set factor levels before plotting. See https://github.com/pwwang/scplotter/issues/29#issuecomment-3009694130 for examples.

highlight A specification for highlighted points:

- NULL (default): no highlighting.
- TRUE: highlight all points (adds a dark outline around every point).
- A character string: a dplyr filter expression (e.g., "clusters == 'Ductal'").
- A character vector: row names to highlight.
- A numeric vector: row indices to highlight.

highlight_alpha
A numeric value in [0, 1] for the transparency of highlighted points. Default is 1.

highlight_size A numeric value for the size of the inner (coloured) highlight point. Default is 1.

highlight_color
A character string for the colour of the outer highlight ring. Default is "black".

highlight_stroke
A numeric value for the thickness of the outer highlight ring (the difference between the outer ring size and highlight_size). Default is 0.8.

add_mark A logical value. If TRUE, group boundaries are drawn around points using ggforce marks. Requires group_by. Only supported in 2D.

mark_type A character string specifying the mark shape. Options: "hull" (convex hull, default), "ellipse", "rect", or "circle".

mark_expand A unit value for the outward expansion of the mark boundary. Passed to ggforce::geom_mark_*(expand = ...). Default is unit(3, "mm").

mark_alpha A numeric value in [0, 1] for the transparency of the mark fill. Default is 0.1.

mark_linetype A numeric value for the line type of the mark boundary. Default is 1 (solid).

stat_by
: A character string naming a column used to compute per-group statistical summary mini-plots embedded at group centroid positions. Only supported with group_by (not features). Only supported in 2D without facet_by.

stat_plot_type
: A character string specifying the mini-plot type. Options: "pie" (default), "ring", "bar", or "line".

stat_plot_size
: A numeric value for the size of the stat mini-plot, expressed as a fraction of the axis range. Default is 0.1.

stat_args
: A list of additional arguments passed to the stat plot function (e.g., list(palette = "Set1")). Default is list(palette = "Set1").

graph
: A specification for network / graph edges to overlay. Sources:

  - A character string starting with "@" (e.g., "@graph"): extracts the attribute named "graph" from attributes(data).
  - A Graph object (e.g., Seurat): coerced to dense matrix via as.matrix().
  - A matrix, data.frame, or dgCMatrix: used directly as the adjacency matrix.
  - Numeric indices or character column names: extracts columns from data. Edges are drawn for non-zero, lower-triangle entries. Requires data to have row names matching the matrix dimnames.

edge_size
: A numeric vector of length 2 specifying the range [min, max] for scale_linewidth_continuous(range = ...) applied to edge widths. Default is c(0.05, 0.5).

edge_alpha
: A numeric value in [0, 1] for the transparency of graph edges. Default is 0.1.

edge_color
: A character string for the colour of graph edges. Default is "grey40".

add_density
: A logical value. If TRUE, a 2D density layer is overlaid. Only supported in 2D.

density_color
: A character string for the colour of the density contour lines. Used when density_filled = FALSE. Default is "grey80".

density_filled
: A logical value. If TRUE, the density is rendered as a filled raster (stat_density_2d(geom = "raster")) instead of contour lines. A separate fill scale is used.

density_filled_palette
: A character string naming the palette for the filled density layer. Default is "Greys".

density_filled_palcolor
: A character vector of specific colours for the filled density palette. Default is NULL (auto-resolved from density_filled_palette).

lineages
: A character vector of column names representing pseudotime / trajectory lineages. Each column is fitted with a LOESS smooth (span = lineages_span, degree = 2) across the 2D embedding, after trimming the top and bottom lineages_trim quantiles. Only supported in 2D without facet_by.

lineages_trim
: A numeric vector of length 2 specifying the lower and upper quantile thresholds [0, 1] for trimming lineage values before LOESS fitting. Default is c(0.01, 0.99).

lineages_span
: A numeric value passed as span to stats::loess() controlling the smoothness of the lineage curve. Smaller values follow the data more closely. Default is 0.75.

lineages_palette
: A character string naming the palette for lineage colours. Default is "Dark2".

lineages_palcolor
: A character vector of specific colours for lineage curves. Default is NULL (auto-resolved from lineages_palette).

lineages_arrow
: A ggplot2 arrow specification applied to the end of lineage paths. Default is arrow(length = unit(0.1, "inches")).

lineages_linewidth
: A numeric value for the width of the lineage curve lines. Default is 1.

lineages_line_bg
: A character string for the colour of the background (wider) stroke drawn behind each lineage curve for improved visibility. Default is "white".

lineages_line_bg_stroke
: A numeric value for the additional width of the background stroke relative to lineages_linewidth. The background line has total width lineages_linewidth + lineages_line_bg_stroke. Default is 0.5.

lineages_whiskers
: A logical value. If TRUE, short line segments connect the smoothed lineage curve to the original data coordinates of the fitted points. Default is FALSE.

lineages_whiskers_linewidth
: A numeric value for the width of the whisker lines. Default is 0.5.

lineages_whiskers_alpha
: A numeric value in [0, 1] for the transparency of the whisker lines. Default is 0.5.

velocity
: A specification for RNA-velocity arrows. Can be:

  - NULL (default): no velocity overlay.
  - A character / integer vector: column names or indices in data for the velocity embedding.
  - A data frame or matrix: the velocity embedding itself (must align with data rows). Only supported in 2D without facet_by.

velocity_plot_type
: A character string specifying the velocity rendering style. Options: "raw" (arrows from embedding), "grid" (grid-based arrows), or "stream" (streamlines). Default is "raw".

velocity_n_neighbors
: A numeric value for the number of neighbours used in the velocity grid computation. Default is NULL (auto).

velocity_density
: A numeric value for the velocity kernel density bandwidth. Default is 1.

velocity_smooth
: A numeric value for the velocity smoothing parameter. Default is 0.5.

velocity_scale
: A numeric value for scaling the velocity arrows. Default is 1.

velocity_min_mass
: A numeric value for the minimum cell mass threshold in velocity grid computation. Default is 1.

velocity_cutoff_perc
: A numeric value for the velocity cutoff percentage. Default is 5.

velocity_group_palette
: A character string naming the palette for velocity group colours (used in "raw" plot type). Default is "Set2".

velocity_group_palcolor
: A character vector of specific colours for velocity groups. Default is NULL (auto-resolved from velocity_group_palette).

arrow_angle
: A numeric value specifying the angle of the arrowheads in degrees. Applied to arrow when plot_type is "raw" or "grid". Default is 20.

arrow_color
: A character string specifying the color of the velocity arrows. For plot_type = "stream", this sets only the arrowhead color. Default is "black".

arrow_alpha
: A numeric value between 0 and 1 specifying the transparency of the velocity arrows. Only used when plot_type = "raw" or "grid"; for plot_type = "stream", use streamline_alpha instead. Default is 1.

streamline_l
: A numeric value specifying the integration length of the streamlines. Passed to geom_streamline as the L parameter. Default is 5.

streamline_minl
: A numeric value specifying the minimum streamline length. Shorter streamlines are not drawn. Passed to geom_streamline as the min.L parameter. Default is 1.

streamline_res
: A numeric value specifying the resolution of the streamline integration. Passed to geom_streamline as the res parameter. Default is 1.

streamline_n
: A numeric value specifying the number of streamlines to draw. Passed to geom_streamline as the n parameter. Default is 15.

streamline_width
: A numeric vector of length 2 specifying the range of line widths for streamlines. Passed to scale_size(range = ...). Only used when streamline_color is NULL. Default is c(0, 0.8).

streamline_alpha
: A numeric value between 0 and 1 specifying the transparency of the velocity streamlines. Default is 1.

streamline_color
: An optional character string specifying a fixed color for streamlines. When NULL (the default), streamlines are colored by velocity magnitude using streamline_palette.

streamline_palette
: A character string specifying the color palette for streamline velocity magnitude. Passed to palette_this. Only used when streamline_color is NULL. Default is "RdYlBu".

streamline_palcolor
: An optional character vector of specific colors for the streamline velocity gradient. If NULL, colors are generated from streamline_palette. Default is NULL.

streamline_bg_color
: A character string specifying the background (outline) color applied to streamlines to create a stroke effect. Default is "white".

streamline_bg_stroke
: A numeric value specifying the additional line width of the background stroke relative to the foreground streamline. Default is 0.5.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

raster
: A logical value. If TRUE, points are rendered via scattermore::geom_scattermore() for efficient rasterised plotting. Default is NULL, which auto-enables when nrow(data) > 1e5.

raster_dpi
: A numeric vector of length 2 [x_dpi, y_dpi] specifying the raster resolution in pixels. Passed to scattermore::geom_scattermore(pixels = ...). Default is c(512, 512). If a single value is provided it is recycled to both dimensions.

hex
: A logical value. If TRUE, points are rendered as hexagonal bins via geom_hex() / stat_summary_hex(). Not supported with highlight. Default is FALSE. Only supported in 2D.

hex_linewidth
: A numeric value for the width of the hexagon boundary lines. Default is 0.5.

hex_count
: A logical value. If TRUE and group_by is set, hex fill alpha is mapped to after_stat(count) so denser bins are more opaque. For features mode hex_count is ignored. Default is !is.null(group_by).

hex_bins
: A numeric value for the number of hex bins along each axis. Passed to geom_hex(bins = ...). Default is 50.

hex_binwidth
: A numeric value for the width of individual hex bins. Passed to geom_hex(binwidth = ...). Takes precedence over hex_bins when set.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.

- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

features — A character vector of the column names to plot as features (continuous colouring). When multiple features are provided and facet_by is not set, the data is pivoted to long format and faceted by feature name.

lower_quantile, upper_quantile — Lower and upper quantiles for the continuous color/fill scale. The actual cutoffs are determined by these quantiles when lower_cutoff and upper_cutoff are NULL. Defaults: lower_quantile = 0, upper_quantile = 0.99.

lower_cutoff, upper_cutoff — Explicit lower and upper cutoffs for the continuous color/fill scale. When NULL (the default), the cutoffs are determined by lower_quantile and upper_quantile via quantile. Values outside the [lower_cutoff, upper_cutoff] range are clamped (winsorized) to the nearest cutoff value.

bg_cutoff — A numeric threshold. Feature values with absolute value below this cutoff are set to NA (and therefore rendered in bg_color). Default is NULL.

color_name — A character string used as the title for the continuous colour bar in feature mode. Default is "".

Value

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE). When dims has 3 elements, a **plotly** object is returned instead.

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE). When dims has 3 elements, a **plotly** object is returned instead.

split_by Workflow (DimPlot)

When split_by is specified, DimPlot() executes the following pipeline:

1. **Argument validation** — validate_common_args() checks the seed and blocks split_by + velocity combinations.

2. **NA / empty normalisation** — `check_keep_na()` / `check_keep_empty()` convert `keep_na` / `keep_empty` to per-column lists.
3. **Theme resolution** — `process_theme()` resolves the theme string to a theme function.
4. **Split column resolution** — `check_columns()` validates `split_by` (force_factor, concat_multi).
5. **Pre-filtering** — `process_keep_na_empty()` removes NA / empty levels from the split column, then `data` is split by `split_by` levels (order preserved). When `graph` references an attribute (`@graph`), the graph matrix is also subset per split.
6. **Per-split parameter resolution** — `check_palette()`, `check_palcolor()`, `check_legend()` resolve palette, palcolor, legend.position, and legend.direction for each split.
7. **Per-split dispatch** — each split is passed to `DimPlotAtomic()` with its resolved parameters. Title defaults to the split level name unless `title` is a function.
8. **Combination** — `combine_plots()` assembles the list of plots via `patchwork::wrap_plots()`, applying `nrow`, `ncol`, `byrow`, `axes`, `axis_titles`, `guides`, and `design`.

split_by Workflow (FeatureDimPlot)

`FeatureDimPlot()` supports two forms of splitting:

A. `split_by = TRUE` **(split by features)**

1. Each feature in `features` is dispatched individually to `DimPlotAtomic()`, producing one plot per feature. The plot title defaults to the feature name.
2. Plots are combined via `combine_plots()` with `split_by = ".features"`.

B. `split_by` **as a column name (split by data column)**

1. Data is split by the named column's levels (same pipeline as DimPlot steps 1–8 above). Graph attribute splitting is supported.

See Also

`VelocityPlot`

Examples

```
data(dim_example)

# basic dim plot
DimPlot(dim_example, group_by = "clusters")
DimPlot(dim_example, group_by = "clusters", theme = "theme_blank")
DimPlot(dim_example, group_by = "clusters", theme = ggplot2::theme_classic,
    theme_args = list(base_size = 16), palette = "seurat")

# raster and highlighting
DimPlot(dim_example, group_by = "clusters", raster = TRUE, raster_dpi = 50)
DimPlot(dim_example, group_by = "clusters", highlight = 1:20,
    highlight_color = "black", highlight_stroke = 2)
DimPlot(dim_example, group_by = "clusters", highlight = TRUE, facet_by = "group",
    theme = "theme_blank")
```

```
# labels
DimPlot(dim_example, group_by = "clusters", label = TRUE,
    label_size = 5, label_bg_r = 0.2)
DimPlot(dim_example, group_by = "clusters", label = TRUE, label_fg = "red",
    label_bg = "yellow", label_size = 5)
DimPlot(dim_example, group_by = "clusters", label = TRUE, label_insitu = TRUE)

# group marks
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE)
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE, mark_linetype = 2)
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE, mark_type = "ellipse")

# density overlays
DimPlot(dim_example, group_by = "clusters", add_density = TRUE)
DimPlot(dim_example, group_by = "clusters", add_density = TRUE, density_filled = TRUE)
DimPlot(dim_example, group_by = "clusters", add_density = TRUE, density_filled = TRUE,
    density_filled_palette = "Blues", highlight = TRUE)

# statistics at group centroids
DimPlot(dim_example, group_by = "clusters", stat_by = "group")
DimPlot(dim_example, group_by = "clusters", stat_by = "group",
    stat_plot_type = "bar", stat_plot_size = 0.06)

# hex bins
DimPlot(dim_example, group_by = "clusters", hex = TRUE)
DimPlot(dim_example, group_by = "clusters", hex = TRUE, hex_bins = 20)
DimPlot(dim_example, group_by = "clusters", hex = TRUE, hex_count = FALSE)

# graph / network edges
DimPlot(dim_example, group_by = "clusters", graph = "@graph", edge_color = "grey80")

# lineages / trajectories
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"))
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"),
    lineages_whiskers = TRUE, lineages_whiskers_linewidth = 0.1)
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"),
    lineages_span = 0.4)

# split_by
DimPlot(dim_example, group_by = "clusters",  split_by = "group",
    palette = list(A = "Paired", B = "Set1"))

# velocity
DimPlot(dim_example, group_by = "clusters", velocity = c("stochasticbasis_1", "stochasticbasis_2"),
    pt_alpha = 0)
DimPlot(dim_example, group_by = "clusters", velocity = 3:4,
    velocity_plot_type = "grid", arrow_alpha = 0.6)
DimPlot(dim_example, group_by = "clusters", velocity = 3:4,
    velocity_plot_type = "stream")

# 3D plots (returns a plotly object)
DimPlot(dim_example, dims = 1:3, group_by = "clusters")
DimPlot(dim_example, dims = 1:3, group_by = "clusters", label = TRUE,
```

```
    label_insitu = TRUE)
DimPlot(dim_example, dims = c("basis_1", "basis_2", "stochasticbasis_1"),
    group_by = "clusters", graph = "@graph", edge_color = "grey80")

# keep_na and keep_empty
dim_example$clusters[dim_example$clusters == "Ductal"] <- NA

DimPlot(dim_example, group_by = "clusters", keep_na = FALSE, keep_empty = TRUE)
DimPlot(dim_example, group_by = "clusters", keep_na = TRUE, keep_empty = TRUE)
DimPlot(dim_example, group_by = "clusters", keep_na = TRUE, keep_empty = FALSE)


data(dim_example)

# single feature
FeatureDimPlot(dim_example, features = "stochasticbasis_1", pt_size = 2)
FeatureDimPlot(dim_example, features = "stochasticbasis_1", pt_size = 2, bg_cutoff = 0)
FeatureDimPlot(dim_example, features = "stochasticbasis_1", raster = TRUE, raster_dpi = 30)

# multiple features (auto-pivoted to long, faceted by feature)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 pt_size = 2)

# single feature with facet_by (facet_by works when only 1 feature)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1"), pt_size = 2,
 facet_by = "group")

# multiple features with split_by for independent layout
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 split_by = "group", nrow = 2)

# highlight and hex
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 highlight = TRUE)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 hex = TRUE, hex_bins = 15)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 hex = TRUE, hex_bins = 15, split_by = "group", palette = list(A = "Reds", B = "Blues"))

# 3D plots (returns a plotly object)
FeatureDimPlot(dim_example, dims = 1:3, features = "stochasticbasis_2", pt_size = 2)
FeatureDimPlot(dim_example, dims = c("basis_1", "basis_2", "stochasticbasis_1"),
 features = "stochasticbasis_2")
```

---

dim_example *An example data for dimensionality reduction plot*

Description

This dataset is generated from the scvelo (scv.datasets.pancreas()) with the scvelo run on the dataset. Then the cell embeddings and velocity embeddings are extracted (200 downsampled), which are the first 4 columns of the data frame. The fifth column is the group identifier (clusters), and the sixth column is a fake grouping variable used to visualize stats, facetting, etc. An attribute "graph" is added to the data frame, which is a square matrix of the cell-cell distances, which is used for the graph (network) on dimensionality reduction plots.

`DotPlot` *Dot Plot, Scatter Plot, and Lollipop Plot*

Description

`DotPlot()` renders a matrix of filled circles (dot plot) where dot size encodes one numeric variable and fill colour encodes another. Either axis can be numeric or factor, enabling four layout combinations:

- **Both axes factor** — a classic dot matrix (e.g. genes × cell types), where each cell is a dot whose size reflects expression magnitude and whose colour reflects a summary statistic.
- **Both axes numeric** — a scatter plot, with dots positioned by x/y coordinates, sized by a third variable, and coloured by a fourth.
- **One numeric, one factor** — a strip plot or (with `lollipop = TRUE`) a lollipop chart.

`LollipopPlot()` is a convenience wrapper that sets `lollipop = TRUE`, producing horizontal bars from the y-axis to each data point, capped by filled dots. It expects a numeric `x` and a factor/character `y`.

Key features:

- **Auto-count:** when `size_by = NULL`, the per-combination observation count is computed automatically.
- **fill_cutoff:** values in `fill_by` matching a threshold expression (e.g. `"< 18"`) are greyed out with a dedicated legend entry.
- **Background stripes:** `add_bg = TRUE` draws alternating background bands along the discrete axis for visual grouping.
- **Border modes:** `border_color` can track the fill gradient (`TRUE`), use a constant colour (`"black"`), or be suppressed (`FALSE`).
- **Colour scale trimming:** `lower_quantile` / `upper_quantile` (or explicit `lower_cutoff` / `upper_cutoff`) trim the continuous fill scale extremes.

`LollipopPlot()` is a convenience wrapper around `DotPlot()` that sets `lollipop = TRUE`. It renders a horizontal bar extending from the y-axis (`x = 0`) to each data point, capped by a filled dot. The bar has a two-layer construction: an outer shadow (black or custom colour) and an inner coloured segment that follows the `fill_by` gradient. Dot size scales by `size_by` (or the per-combination observation count when `size_by = NULL`).

Expects `x` to be a numeric column and `y` to be a factor or character column.

Usage

```
DotPlot(
  data,
  x,
  y,
  x_sep = "_",
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  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"),
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_cutoff_by = NULL,
  palreverse = FALSE,
  size_min = 1,
  size_max = 10,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
```

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

LollipopPlot(
  data,
  x,
  y,
  y_sep = NULL,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  fill_cutoff_by = NULL,
  palreverse = FALSE,
  size_min = 1,
  size_max = 10,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
```

```
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string naming the column for the x-axis. Must be a numeric column (bars extend from 0 to the data value).
y	A character string naming the column for the y-axis. Must be a factor or character column (each level gets a lollipop bar).
x_sep	A character string used to join multiple x column values into a single factor level. Only used when x is non-numeric and multiple columns are provided. Default: "_".
y_sep	A character string used to join multiple y column values into a single factor level. Only used when y is non-numeric and multiple columns are provided. Default: "_".
flip	A logical value. If TRUE, the x and y axes are swapped via coord_flip(). Dimension calculation accounts for the flip. Default: FALSE.
split_by	The column(s) to split data by and generate separate plots for each level. The split column is processed for keep_na / keep_empty before splitting.
split_by_sep	A character string used to concatenate multiple split_by column values. Default: "_".
size_name	A character string for the size legend title. When NULL (the default), the size_by column name is used.

fill_name — A character string for the fill colour-bar legend title. When NULL (the default), the fill_by column name is used.

fill_cutoff_name — A character string for the fill cutoff legend title (shown when fill_cutoff is active). Defaults to "<fill_by> <fill_cutoff>", e.g. "mpg < 18".

add_bg — A logical value. If TRUE, alternating background stripes are drawn behind the points via bg_layer(). The striped axis is determined by bg_direction. Requires the striped axis to be non-numeric. Default: FALSE.

bg_palette — A character string specifying the palette for the background stripe colours. Passed to bg_layer(). Default: "stripe".

bg_palcolor — A character vector of colours for the background stripes. Passed to bg_layer(). When NULL (default), colours are derived from bg_palette.

bg_alpha — A numeric value in [0, 1] for the transparency of the background stripes. Default: 0.2.

bg_direction — A character string specifying which axis receives the alternating background stripes. "vertical" (default) stripes by x levels; "horizontal" stripes by y levels. Abbreviations "v" and "h" are also accepted.

size_by — A character string naming a numeric column whose values control dot size. When NULL (the default), the per-combination observation count is computed automatically (via dplyr::summarise(n = n())) and used as the size variable. If fill_by is also present, the first value of fill_by per combination is retained with a warning. A single numeric value is also accepted and sets a constant dot size (used by ScatterPlot).

fill_by — A character string naming a numeric column whose values control the fill colour of the dots (and lollipop inner bars). A continuous gradient from palette is applied via scale_fill_gradientn(). When NULL (the default), all dots are filled with a single constant colour from the middle of the palette.

fill_cutoff — A string expression specifying which values of fill_by to grey out. Format: an operator followed by a number, e.g. "< 18", "<= 18", "> 18", or ">= 18". Values matching the condition are set to NA and rendered in grey ("grey80"), while the rest are coloured by the fill gradient. The operator determines which side of the threshold is greyed out, independent of palreverse. A numeric value is also accepted as shorthand for "<" (e.g. 18 is equivalent to "< 18"). Requires fill_by to be set.

fill_cutoff_by — A character string giving the column used to decide which dots are greyed out. Defaults to fill_by, so the cutoff is applied to the column that also drives the fill colour. Set it to another column to grey out dots based on a different measure while still colouring them by fill_by.

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

size_min — A numeric value for the smallest dot size in the scale_size(range = c(size_min, size_max)) range. Default: 1.

size_max — A numeric value for the largest dot size in the scale_size(range = c(size_min, size_max)) range. Default: 10.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha
: A numeric value specifying the transparency of the plot.

border_color
: Controls the dot border colour and lollipop outer-shadow appearance:
  - TRUE — dot borders and lollipop inner bars follow the fill_by gradient via scale_color_gradientn(); lollipop outer shadow is black.
  - "black" (default) — constant black borders on dots and black outer shadow on lollipop bars.
  - A colour string (e.g. "red", "#FF0000") — constant colour for both dot borders and lollipop outer shadows.
  - FALSE — no dot borders and no lollipop outer shadow (the inner coloured bars remain visible in lollipop mode).

border_size
: A numeric value for the stroke width of dot borders and the base linewidth of lollipop bars. In lollipop mode, the outer shadow uses border_size * 4 and the inner bar uses border_size * 2. Default: 0.5.

border_alpha
: A numeric value in [0, 1] controlling the transparency of dot borders and lollipop bar segments. Default: 1.

lower_quantile, upper_quantile
: Lower and upper quantiles for the continuous color/fill scale. The actual cutoffs are determined by these quantiles when lower_cutoff and upper_cutoff are NULL. Defaults: lower_quantile = 0, upper_quantile = 0.99.

lower_cutoff, upper_cutoff
: Explicit lower and upper cutoffs for the continuous color/fill scale. When NULL (the default), the cutoffs are determined by lower_quantile and upper_quantile via quantile. Values outside the [lower_cutoff, upper_cutoff] range are clamped (winsorized) to the nearest cutoff value.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

x_text_angle
: A numeric value specifying the angle of the x-axis text.

seed
: The random seed for reproducibility. Passed to validate_common_args(). Default: 8525.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

keep_na A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

- `FALSE` (default): Drop empty factor levels from the data before plotting.
- `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
- `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

combine A logical value. If `TRUE` (the default), the list of per-split plots is combined into a single `patchwork` object. If `FALSE`, returns the raw list.

nrow, ncol, byrow
: Integers controlling the layout of combined plots via `patchwork::wrap_plots()`. `byrow = TRUE` (default) fills the layout row-wise.

axes, axis_titles
: Strings controlling how axes and axis titles are handled across combined plots. Passed to `combine_plots()`. See `?patchwork::wrap_plots` for options (`"keep"`, `"collect"`, `"collect_x"`, `"collect_y"`).

guides A string controlling guide collection across combined plots. Passed to `combine_plots()`.

design A custom layout specification for combined plots. Passed to `combine_plots()`. When specified, `nrow`, `ncol`, and `byrow` are ignored.

... Additional arguments.

Value

A ggplot object (single plot), a patchwork object (when combine = TRUE with split_by), or a list of ggplot objects (when combine = FALSE).

A ggplot object (single plot), a patchwork object (when combine = TRUE with split_by), or a list of ggplot objects (when combine = FALSE).

split_by Workflow (DotPlot)

When split_by is provided, the following pipeline executes:

1. **Column validation** — check_columns() resolves split_by (force_factor, allow_multi, concat_multi).
2. **NA / empty pre-processing** — process_keep_na_empty() handles keep_na / keep_empty for the split column before splitting, then removes the split column from the per-split keep_na/keep_empty lists.
3. **Data splitting** — splits data by split_by levels (preserving factor level order).
4. **Per-split palette / colour** — check_palette() and check_palcolor() resolve per-split palette and colour overrides.
5. **Per-split legend** — check_legend() resolves legend.position and legend.direction per split.
6. **Per-split title** — when title is a function, it receives the default title (the split level name) and can return a custom string; otherwise title %||% split_level is used.
7. **Dispatch** — each split subset is passed to DotPlotAtomic (with lollipop = FALSE).
8. **Combination** — combine_plots() assembles the list of plots via patchwork::wrap_plots, honouring nrow/ncol/byrow/design.

split_by Workflow (LollipopPlot)

Same pipeline as DotPlot above, but dispatches to DotPlotAtomic with lollipop = TRUE.

Examples

```
mtcars <- datasets::mtcars
mtcars$carb <- factor(mtcars$carb)
mtcars$gear <- factor(mtcars$gear)

# --- Basic dot plot (factor × factor, size + fill) ---
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")

# --- Background stripes ---
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", add_bg = TRUE)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", add_bg = TRUE,
        bg_direction = "h")
```

```
# --- Faceting ---
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", facet_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", facet_by = "cyl",
        facet_scales = "free_x")

# --- split_by ---
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", split_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18", split_by = "cyl",
        palette = list("4" = "Set1", "6" = "Paired", "8" = "Reds"))

# --- Scatter plot (both axes numeric) ---
DotPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18",
        fill_cutoff_name = "Small mpgs")

# --- keep_na and keep_empty ---
mtcars$carb[mtcars$carb == "1"] <- NA
mtcars$gear[mtcars$gear == "3"] <- NA
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = "< 18",
        keep_na = TRUE, keep_empty = TRUE)

# --- Border customization ---
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", border_color = "red", border_size = 2)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", border_color = TRUE, border_size = 1.5,
        border_alpha = 0.5)
DotPlot(mtcars, x = "carb", y = "gear",
        fill_by = "mpg", border_color = FALSE)

# --- Colour scale trimming ---
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", lower_quantile = 0.05, upper_quantile = 0.95)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", lower_cutoff = 15, upper_cutoff = 25)


mtcars <- datasets::mtcars

# --- Basic lollipop ---
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg")

# --- Faceting ---
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg", fill_cutoff = "< 18", facet_by = "cyl",
             facet_scales = "free_y")
```

```
# --- split_by ---
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             split_by = "vs", palette = list("0" = "Reds", "1" = "Blues"))

# --- Border customization ---
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg", border_color = "red", border_size = 2)
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg", border_color = TRUE, border_size = 1.5,
             border_alpha = 0.5)
LollipopPlot(mtcars, x = "qsec", y = "drat",
             fill_by = "mpg", border_color = FALSE)
```

`element_textbox` *Theme element that add a box to the text*

Description

Code grabbed from the `ggtext` package. See the original code at: https://github.com/wilkelab/ggtext This is used to create a text box around the text, primarily to be used in `CorPairsPlot`.

Usage

```
element_textbox(
  family = NULL,
  face = NULL,
  size = NULL,
  colour = NULL,
  fill = NULL,
  box.colour = NULL,
  linetype = NULL,
  linewidth = NULL,
  hjust = NULL,
  vjust = NULL,
  halign = NULL,
  valign = NULL,
  lineheight = NULL,
  margin = NULL,
  padding = NULL,
  width = NULL,
  height = NULL,
  minwidth = NULL,
  maxwidth = NULL,
  minheight = NULL,
  maxheight = NULL,
  r = NULL,
```

```
  orientation = NULL,
  color = NULL,
  box.color = NULL,
  debug = FALSE,
  inherit.blank = FALSE
)

## S3 method for class 'element_textbox'
element_grob(
  element,
  label = "",
  x = NULL,
  y = NULL,
  family = NULL,
  face = NULL,
  colour = NULL,
  size = NULL,
  hjust = NULL,
  vjust = NULL,
  lineheight = NULL,
  margin = NULL,
  ...
)
```

Arguments

`family`	Font family
`face`	Font face
`size`	Font size (in pt)
`colour`, `color`	Text color
`fill`	Fill color of the enclosing box
`box.colour`, `box.color`	Line color of the enclosing box (if different from the text color)
`linetype`	Line type of the enclosing box (like `lty` in base R)
`linewidth`	Line width of the enclosing box (measured in mm, just like `size` in `ggplot2::element_line()`).
`hjust`	Horizontal justification
`vjust`	Vertical justification
`halign`	Horizontal justification
`valign`	Vertical justification
`lineheight`	Line height, in multiples of the font size
`padding`, `margin`	Padding and margins around the text box. See `gridtext::textbox_grob()` for details.
`width`, `height`	Unit objects specifying the width and height of the textbox, as in `gridtext::textbox_grob()`.

`minwidth`, `minheight`, `maxwidth`, `maxheight`
: Min and max values for width and height. Set to NULL to impose neither a minimum nor a maximum.

`r`
: Unit value specifying the corner radius of the box

`orientation`
: Orientation of the text box. See `gridtext::textbox_grob()` for details.

`debug`
: Not implemented.

`inherit.blank`
: See `ggplot2::margin()` for details.

`element`
: A theme element created by `element_textbox()`.

`label`
: Text to display in the textbox.

`x`, `y`
: Position of the textbox.

`...`
: Other arguments passed to `gridtext::textbox_grob()`.

Value

A ggplot2 theme element that can be used inside a `ggplot2::theme()` call.

`EnrichMap` *Enrichment Map and Enrichment Network*

Description

`EnrichMap` draws an enrichment map – a gene-set similarity network where each node is an enriched term, node size encodes the number of associated genes, node fill colour encodes cluster membership (detected via igraph community detection), and edge thickness encodes the number of overlapping genes between term pairs. The plot uses a force-directed layout to arrange terms, and ggforce hull annotations group terms into clusters. Keyword or term-description labels appear in the legend.

`EnrichNetwork` draws an enrichment network – a term-gene bipartite graph where term nodes are shown as numbered circles and gene nodes as labelled rectangles. Gene node colours are blended from the colours of all terms they belong to. A force-directed layout positions the nodes, with optional overlap adjustment for better readability.

Both functions accept enrichment results from clusterProfiler or Enrichr (the latter is auto-detected and preprocessed via `prepare_enrichr_result()`).

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),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  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,
  ...
)

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_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  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 enrichment results in clusterProfiler format (see EnrichMap for the expected columns). 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 source. Use split_by = "Database" to plot them side by side.
in_form	A character string specifying the input format. When "auto" (default), the function infers the format from the column names. Other options are "clusterProfiler", "clusterprofiler", and "enrichr".
split_by	The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.
split_by_sep	The separator for multiple split_by columns. See split_by
top_term	An integer specifying the maximum number of terms to include. Terms are ranked by metric (ascending). Default 100.
metric	A character string specifying the significance metric used for top-term selection and node scoring: "p.adjust" (default) or "pvalue". The value is transformed as $-\log_{10}(metric)$.

layout	A character string naming the igraph layout algorithm. Built-in shortcuts: "circle", "tree", "grid". Otherwise, the suffix passed to layout_with_<layout> in igraph (e.g. "fr" for Fruchterman-Reingold, "kk" for Kamada-Kawai). Default "fr".
minchar	An integer specifying the minimum character length for words to be included as keywords when show_keyword = TRUE. Default 2.
cluster	A character string naming the igraph community detection algorithm. The suffix passed to cluster_<cluster> in igraph (e.g. "fast_greedy", "walktrap", "edge_betweenness", "infomap"). Default "fast_greedy".
show_keyword	A logical value. When TRUE, the Description text is tokenized and the most significant words per cluster are shown as keywords. When FALSE (default), the original term descriptions are used as labels.
nlabel	An integer specifying the number of keywords or term descriptions to show per cluster in the legend labels. Default 4.
character_width	An integer specifying the maximum width (in characters) at which keyword labels are wrapped via strwrap(width = character_width). Default 50.
mark	A character string naming the ggforce hull function. One of "ellipse" (default), "rect", "circle", or "text" – passed as the suffix to geom_mark_<mark>.
label	A character string specifying what information to display in the legend labels. Either "term" (default; shows top term descriptions/keywords per cluster) or "feature" (shows top gene symbols per cluster).
labelsize	A numeric value specifying the font size of the cluster labels drawn by the ggforce mark layer. Default 5.
expand	The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

alpha A numeric value specifying the transparency of the plot.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

seed The random seed to use. Default is 8525.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

axes A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.
layoutadjust	A logical value. When TRUE (default), applies adjust_network_layout() after the initial layout to reduce node overlap based on label width and a repulsion simulation.
adjscale	A numeric value controlling the scale of the layout adjustment. Passed as the scale argument to adjust_network_layout(). Default 60.
adjiter	A numeric value controlling the number of iterations for the layout adjustment. Passed as the iter argument to adjust_network_layout(). Default 100.
blendmode	A character string specifying how gene colours are computed from the colours of the terms they belong to. One of "blend" (default), "average", "multiply", or "screen". Passed to blend_colors().

Value

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE).

A ggplot object (single plot), a patchwork / wrap_plots object (when split_by is provided and combine = TRUE), or a list of ggplot objects (when split_by is provided and combine = FALSE).

split_by Workflow (EnrichMap)

When split_by is provided, EnrichMap() executes the following pipeline:

1. **Argument validation** – validate_common_args() checks the seed.
2. **Input format detection** – match.arg() resolves in_form; "auto" mode infers the format from column names.
3. **Enrichr preprocessing** – when format is "enrichr", calls prepare_enrichr_result() to rename columns and infer GeneRatio/BgRatio.
4. **Split column resolution** – check_columns() validates split_by (force_factor, allow_multi, concat_multi).
5. **Data splitting** – splits data by split_by levels, preserving factor level order.
6. **Per-split palette/colour** – check_palette() and check_palcolor() resolve per-split palette and colour overrides.
7. **Per-split legend** – check_legend() resolves legend.position and legend.direction per split.
8. **Per-split title** – when title is a function, it receives the default title (the split level name); otherwise title %||% split_level is used.
9. **Dispatch** – each split subset is passed to EnrichMapAtomic with its resolved parameters.
10. **Combination** – combine_plots() assembles the list of plots via patchwork::wrap_plots, honouring nrow/ncol/byrow/axes/ axis_titles/guides/design.

split_by Workflow (EnrichNetwork)

When `split_by` is provided, `EnrichNetwork()` executes the same pipeline as `EnrichMap()` above, but dispatches each split subset to `EnrichNetworkAtomic`.

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(enrich_example, top_term = 5)
```

---

enrich_example *An example of clusterProfiler enrichment result*

Description

An example of clusterProfiler enrichment result

Examples

```
## Not run:
if (interactive()) {
  data(geneList, package="DOSE")
  de <- names(geneList)[abs(geneList) > 1.5]
 enrich_example <- clusterProfiler::enrichPathway(gene=de, pvalueCutoff = 0.05, readable=TRUE)
  enrich_example <- as.data.frame(enrich_example)
}

## End(Not run)
```

enrich_multidb_example *An example of clusterProfiler enrichment result with multiple databases*

Description

An example of clusterProfiler enrichment result with multiple databases

Examples

```
## Not run:
if (interactive()) {
  data(enrich_example, package="plotthis")
  enrich_example$Database <- "DB1"
  enrich_example2 <- enrich_example
  enrich_example2$Database <- "DB2"
  enrich_example2$ID <- paste0(enrich_example2$ID, "_DB2")
  enrich_example2$Description <- paste0(enrich_example2$Description, " (DB2)")
  enrich_multidb_example <- rbind(enrich_example, enrich_example2)
}

## End(Not run)
```

GSEASummaryPlot *GSEA summary dot plot*

Description

Produces a summary dot plot of GSEA (Gene Set Enrichment Analysis) results. Each row represents a gene set (term), positioned along the x-axis by its Normalized Enrichment Score (NES). Dot colour encodes the significance level (typically `-log10(p.adjust)`) on a continuous gradient, and each row includes a miniature line plot showing the gene ranks or running enrichment score for that term's gene set.

The function supports both `DOSE` and `fgsea` package output formats via the `in_form` parameter. Terms can be ranked and selected by a significance metric (`top_term`, `metric`), with non-significant terms rendered in grey. The per-term line plots can show either the raw preranked gene statistics (`line_by = "prerank"`) or the running enrichment score (`line_by = "running_score"`).

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_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  ...
)

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,
  ...
)
```

Arguments

data	A data frame.
in_form	The format of the input data. See GSEASummaryPlot for details.
gene_ranks	A named numeric vector of gene-level rank statistics, with gene identifiers as names. Used to construct the per-term line plots. If a character string starting with "@", the attribute of data with that name (minus the "@") is used as the gene ranks vector.
gene_sets	A named list of gene sets. Each name must correspond to an ID in data, and each element is a character vector of gene identifiers. A GSEA ridge plot is generated for each gene set in the list. If you only want to plot a subset of gene sets, subset the list before passing it to this function. If a character string starting with "@", the attribute of data with that name (minus the "@") is used.
top_term	Integer specifying the number of top terms to display, ranked by metric. If NULL, all terms are shown.
metric	Character string specifying the column name used to rank terms and assess significance. Typically "p.adjust" or "pvalue". Terms are ranked by this column (ascending, lower is better) when top_term is set. The same column is transformed to -log10(metric) for the colour gradient.
cutoff	Numeric threshold for the metric column. Terms with values below this cutoff are coloured on a gradient; terms above are drawn in grey ("grey80") and labelled as insignificant via nonsig_name. Default is 0.05. If NULL, all terms are treated as significant.
character_width	Integer specifying the maximum character width for wrapping term descriptions on the y-axis. Default is 50.
line_plot_size	Numeric controlling the size of the per-term miniature enrichment plots embedded in each row. Expressed as a fraction of the plot panel dimensions. Default is 0.25.
metric_name	Character string for the colour bar legend title. Defaults to the value of metric.
nonsig_name	Character string for the legend entry label used for non-significant terms. Default is "Insignificant".
linewidth	Numeric specifying the line width within the per-term miniature enrichment plots. Default is 0.2.
line_by	The method used to compute the per-term line plots:

- "prerank" (default): Use the gene ranks as the bar heights (raw ranking metric).
- "running_score": Use the running enrichment score computed by gsea_running_score().

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

alpha — A numeric value specifying the transparency of the plot.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

seed — A numeric seed for reproducibility. Passed to validate_common_args().

... — Additional arguments.

gs — Character vector of gene set IDs to plot. If NULL (default), all gene sets in gene_sets that appear in data$ID are plotted.

sample_coregenes — Logical; if TRUE, core enrichment genes are sampled randomly for labelling. If FALSE (default), the first n_coregenes core enrichment genes are used.

line_width — Numeric specifying the line width for the running enrichment score curve. Default is 1.5.

line_alpha — Numeric alpha transparency for the running score line and hit indicator bars. Default is 1.

line_color — Character string specifying the colour of the running enrichment score line. Default is "#6BB82D".

n_coregenes — Integer specifying the number of core enrichment genes to label on the running score plot. Default is 10. Ignored when genes_label is provided.

genes_label	Character vector of specific gene names to label on the running score plot. When provided, n_coregenes is ignored.
label_fg	Character string specifying the text colour of gene labels. Default is "black".
label_bg	Character string specifying the background colour of gene labels. Default is "white".
label_bg_r	Numeric specifying the corner radius of the label background. Default is 0.1.
label_size	Numeric specifying the font size of the label text. Default is 4.
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual patchwork objects (one per gene set).
ncol, nrow	Integer number of columns / rows for the combined layout (passed to combine_plots()).
byrow	Logical; fill the combined layout by row. Default TRUE (passed to combine_plots()).
axes	A character string specifying how axes should be treated across the combined layout (passed to combine_plots()).
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides (legends) should be collected across panels (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).

Value

A ggplot object with height and width attributes (in inches) attached.

A patchwork object when combine = TRUE, or a named list of patchwork objects when combine = FALSE. Each individual plot has height and width attributes in inches.

Examples

```
data(gsea_example)

# Default summary dot plot with preranked gene statistics
GSEASummaryPlot(gsea_example)

# Use running enrichment score for per-term line plots
GSEASummaryPlot(gsea_example, line_by = "running_score")

# Raise the significance cutoff (all terms are coloured)
GSEASummaryPlot(gsea_example, cutoff = 0.01)


data(gsea_example)

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

# Multiple gene sets arranged in a grid
GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1:4])
```

gsea_example *An example of GSEA result from fgsea package*

Description

An example of GSEA result from fgsea package

Examples

```
## Not run:
if (interactive()) {
 set.seed(1234)
 data(geneList, package="DOSE")
 gsea_example <- DOSE::gseDO(geneList)
 gene_ranks <- gsea_example@geneList
 gene_sets <- gsea_example@geneSets
 gsea_example_pos <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES > 0, ]
 gsea_example_neg <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES < 0, ]
 gsea_example <- rbind(
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ],
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ]
 )

 attr(gsea_example, "gene_ranks") <- gene_ranks
 attr(gsea_example, "gene_sets") <- gene_sets[gsea_example$ID]
}

## End(Not run)
```

Heatmap *Heatmap*

Description

Draw a heatmap to visualise data in matrix form. This is the public, exported interface — it accepts data in multiple input formats (matrix, wide, or long), preprocesses it via process_heatmap_data, and delegates to HeatmapAtomic for rendering. Commonly used in biology to visualise gene expression, but applicable to any matrix-structured data.

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,
    keep_na = FALSE,
    keep_empty = FALSE,
    rows_orderby = NULL,
    columns_orderby = NULL,
    columns_name = NULL,
    columns_split_name = NULL,
    rows_name = NULL,
    rows_split_name = NULL,
    palette = "RdBu",
    palcolor = NULL,
    palreverse = FALSE,
    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 = 1,
    label = identity,
    label_size = 10,
    label_color = "black",
    label_name = "label",
    mark = identity,
    mark_color = "black",
    mark_size = 1,
    mark_name = "mark",
    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",
  cluster_columns = NULL,
  cluster_rows = NULL,
  show_row_names = NULL,
  show_column_names = NULL,
  border = TRUE,
  title = NULL,
  column_title = NULL,
  row_title = NULL,
  na_col = "grey85",
  row_names_side = "right",
  column_names_side = "bottom",
  row_annotation = NULL,
  row_annotation_side = NULL,
  row_annotation_palette = NULL,
  row_annotation_palcolor = NULL,
  row_annotation_type = NULL,
  row_annotation_params = NULL,
  row_annotation_agg = NULL,
  column_annotation = NULL,
  column_annotation_side = NULL,
  column_annotation_palette = NULL,
  column_annotation_palcolor = NULL,
  column_annotation_type = NULL,
  column_annotation_params = NULL,
  column_annotation_agg = NULL,
  flip = FALSE,
  alpha = 1,
  seed = 8525,
  padding = 15,
  base_size = 1,
  aspect.ratio = NULL,
  draw_opts = list(),
  layer_fun_callback = NULL,
 cell_type = c("tile", "bars", "label", "mark", "label+mark", "mark+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. When a matrix, it is melted to long format internally (requires row and column names).
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 used to fill missing cells in the matrix. Default NA. Missing values prevent clustering when not filled.
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.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

rows_orderby
: A expression (in character) to specify how to order rows. It will be evaluated in the context of the data frame used for rows (after grouping by rows_split_by and rows_by). The expression should return a vector of the same length as the number of rows in the data frame. The default is NULL, which means no specific ordering. Can't be used with cluster_rows = TRUE. This is applied before renaming rows_by to rows_name.

columns_orderby
: A expression (in character) to specify how to order columns. It will be evaluated in the context of the data frame used for columns (after grouping by columns split_by and columns_by). The expression should return a vector of the same length as the number of rows in the data frame. The default is NULL, which means no specific ordering. Can't be used with cluster_columns = TRUE. This is applied before renaming columns_by to columns_name.

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 naming a palette (see show_palettes) or a character vector of colours for the main heatmap colour scale. Default "RdBu".

palcolor — A custom colour vector overriding palette.

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

pie_size_name — Legend title for the pie size.

pie_size — A numeric value or function returning the pie radius. When a function, it receives the count of groups in the pie.

pie_values — A function or string (convertible via match.arg) to compute the value represented by each pie slice. Default "length" counts observations per group.

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, pie_palcolor — Palette and custom colours for pie slice fill colours.

bars_sample — Fraction of each cell's observations ($0 < x <= 1$; 1 uses all data) or a whole count > 1 sampled per cell when cell_type = "bars". Column widths are proportional to the number of bars drawn per column, so a numeric count gives equal column widths and a fraction gives widths proportional to the column data sizes. Proportional widths are skipped when columns_split_by is given, and column clustering is disabled in proportional mode. Default 1 (all data).

label — A function to compute text labels when cell_type = "label" (or "label+mark"). Receives the aggregated value for a cell and optionally row/column indices and names. See below for the full dispatch contract.

label_size — Default point size for label text (used as fallback when the label function does not return a size field).

label_color — Default colour for label text (fallback).

label_name — Legend title for the label colour scale. The legend is shown automatically when the label function returns a legend field for at least one cell.

mark — A function to compute mark symbols when cell_type = "mark" (or "label+mark"). Same dispatch contract as label.

mark_color — Default mark colour (fallback).

mark_size — Default mark stroke width (lwd) in pt (fallback).

mark_name — Legend title for the mark colour scale.

violin_fill — A character vector of colours to use as fill for violin plots when cell_type = "violin". If NULL, the annotation colour is used.

boxplot_fill — A character vector of colours to use as fill for boxplots when cell_type = "boxplot". If NULL, the annotation colour is used.

dot_size — Dot size when cell_type = "dot". Can be a numeric value or a function.

dot_size_name Legend title for the dot size.

legend_items A named numeric vector specifying custom legend entries for the main colour scale. Names become the displayed labels.

legend_discrete
Logical; if TRUE, treat the main colour scale as discrete.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

lower_quantile, upper_quantile, lower_cutoff, upper_cutoff
Quantile or explicit cutoffs for clipping the colour scale. Applied to aggregated values for tile / label cell types; applied to raw values for bars / violin / boxplot types.

add_bg Logical; if TRUE, add a background fill behind non-tile cell types. Not used for cell_type = "tile" or "bars".

bg_alpha Numeric in $[0, 1]$ for background transparency.

add_reticle Logical; if TRUE, draw a reticle (crosshair pattern) over the heatmap.

reticle_color Colour for the reticle lines.

cluster_columns
Logical; cluster the columns. If TRUE and columns_split_by is provided, clustering is applied within each split group.

cluster_rows Logical; cluster the rows. If TRUE and rows_split_by is provided, clustering is applied within each split group.

show_row_names Logical or character vector; show row names. If TRUE, the legend of the row group annotation is hidden. FALSE only hides the in-place names. Alternatively, a character vector combining the display modes "inplace" (in-place names, same as TRUE), "legend" (row group annotation with legend), "simple" (row group annotation without legend), "anno" (alias "annotation"; label annotation showing the row names as text) and "none" (no names, annotation, or legend). Modes are lower-priority defaults: explicit row_annotation entries (e.g. .row) still win.

show_column_names
Logical or character vector; show column names. See show_row_names for the display modes.

border A logical value indicating whether to draw borders around the heatmap. If TRUE, slice borders are also drawn. Default TRUE.

title The global (column) title of the heatmap.

column_title Character string/vector used as the column group annotation title. A character vector consisting entirely of the display modes "inplace" (per-slice split titles), "legend" (split annotation with legend, no title), "simple" (split annotation, no legend/title), "anno" (alias "annotation"; label block annotation) and "none" (no title, annotation, or legend) controls how the split titles are displayed. A literal title identical to a mode keyword is interpreted as a mode.

row_title Character string/vector used as the row group annotation title. See column_title for the display modes.

na_col Colour for NA cells. Default "grey85".

row_names_side Side for row names. Default "right".

column_names_side
Side for column names. Default "bottom".

row_annotation A structured list specifying row annotations. Same format as column_annotation. Sides default to "left". Aliases: .row/.rows for rows_by, .row.split/.rows.split for rows_split_by.

row_annotation_side
Deprecated: use row_annotation with the side sub-key instead.

row_annotation_palette
Deprecated: use row_annotation with the palette sub-key instead.

row_annotation_palcolor
Deprecated: use row_annotation with the palcolor sub-key instead.

row_annotation_type
Deprecated: use row_annotation with the type sub-key instead.

row_annotation_params
Deprecated: use row_annotation with the params sub-key instead.

row_annotation_agg
Deprecated: use row_annotation with the agg sub-key instead.

column_annotation
A structured list specifying column annotations. Each entry is a named list with sub-keys:

col Column name in data supplying the annotation values. If omitted, the entry name is used as the column name.

side "top" or "bottom".

palette Palette name (see show_palettes).

palcolor Custom colour vector overriding palette.

type Annotation type: "auto", "simple", "pie", "ring", "bar", "violin", "boxplot", "density", "label", "rownames", "points", "lines". For split annotations, "rownames" (row splits) / "colnames" (column splits) is like "label" but labels each split block with the concatenated row/column names; "names" and "dimnames" are aliases. params$sep (default " ") controls the separator and params$wrap_by the number of names per line. For name annotations (the built-in rows_by/columns_by annotation), "rownames"/"colnames" renders the row/column names as text labels (this is what show_row_names = "anno" uses).

params A list of additional parameters passed to the annotation constructor. FALSE disables the annotation. $show_legend controls legend visibility. See HeatmapAnnotation.

agg A function to aggregate values for the annotation.

name Display name for the annotation, shown next to the annotation bar and used as the legend title. FALSE hides the displayed name. Defaults to the annotation key (column name).

Shortcuts:

- `column_annotation = list(Score = "score")` is short for `list(Score = list(col = "score"))`.
- `column_annotation = TRUE` enables annotations with defaults. `FALSE` disables all column annotations.

Special keys:

- `.default` — default values inherited by all entries. `params` is merged recursively; other keys are inherited only when the entry does not already specify them.
- `.col` / `.cols` / `.column` / `.columns` — alias for `columns_by` (the built-in name annotation).
- `.col.split` / `.cols.split` / `.column.split` / `.columns.split` — alias for `columns_split_by` (the built-in split annotation).
- `.row` / `.rows` — alias for `rows_by`.
- `.row.split` / `.rows.split` — alias for `rows_split_by`.

`column_annotation_side`

Deprecated: use `column_annotation` with the `side` sub-key instead.

`column_annotation_palette`

Deprecated: use `column_annotation` with the `palette` sub-key instead.

`column_annotation_palcolor`

Deprecated: use `column_annotation` with the `palcolor` sub-key instead.

`column_annotation_type`

Deprecated: use `column_annotation` with the `type` sub-key instead.

`column_annotation_params`

Deprecated: use `column_annotation` with the `params` sub-key instead.

`column_annotation_agg`

Deprecated: use `column_annotation` with the `agg` sub-key instead.

`flip` Logical; if `TRUE`, swap rows and columns transparently. The caller does not need to swap row- and column-related arguments manually.

`alpha` Alpha transparency for heatmap cells in $[0, 1]$.

`seed` The random seed to use. Default is 8525.

`padding` Padding around the heatmap in CSS order (top, right, bottom, left). Supports 1–4 values. Default 15 (mm). Note that this is different from `ComplexHeatmap::draw()`'s `padding` argument which uses bottom-left-top-right order.

`base_size` A positive numeric scalar used as a scaling factor for the overall heatmap size. Default 1 (no scaling). Values > 1 enlarge all cell dimensions proportionally.

`aspect.ratio` Height-to-width ratio of a single heatmap cell. When `NULL` (default), sensible per-`cell_type` defaults are used: 1 for `tile`/`label`/`dot`, 0.5 for `bars`, and 2 for `violin`/`boxplot`/`pie`. The ratio is constrained by the overall plot dimensions.

`draw_opts` A named list of additional arguments passed to `draw,HeatmapList-method`. Internally managed arguments take precedence.

`layer_fun_callback`

A function to add custom graphical layers on top of each heatmap cell. Receives `j`, `i`, `x`, `y`, `w`, `h`, `fill`, `sr`, `sc`. See `Heatmap` for details.

`cell_type` The type of cell to render. One of `"tile"` (default), `"bars"`, `"label"`, `"mark"`, `"label+mark"` (or `"mark+label"`), `"dot"`, `"violin"`, `"boxplot"`, `"pie"`. See the **Cell types** section for details.

`cell_agg` A function to aggregate values within each cell when `cell_type = "tile"` or `"label"`. Default is `mean`.

`combine` Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow` A numeric value specifying the number of rows in the facet.

`ncol` A numeric value specifying the number of columns in the facet.

`byrow` A logical value indicating whether to fill the plots by row.

`axes` A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments passed to `HeatmapAtomic`, which in turn forwards them to `Heatmap`.

Value

A `patchwork` object (class `wrap_plots`) with `height` and `width` attributes (in inches). When `combine = FALSE`, a named list of such objects, one per `split_by` level.

Input formats

The `in_form` parameter controls how the input `data` is interpreted:

- `"auto"` (default) — detects the format automatically.
- `"matrix"` — `data` is a matrix with row and column names. It is melted to long form internally.
- `"wide-rows"` — each row is a feature, columns are samples.
- `"wide-columns"` — each column is a feature, rows are samples.
- `"long"` — tidy/long format with one observation per row.

Split-by support

When `split_by` is provided, the data is partitioned into subsets and an independent heatmap is produced for each level. Results are combined via `wrap_plots` according to `nrow`, `ncol`, `byrow`, and `design`. Per-split `palette`, `palcolor`, `legend.position`, and `legend.direction` can be specified as named lists keyed by split level.

See Also

`HeatmapAtomic`, `LinkedHeatmap`, `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 = list(col = "group", palette = "Spectral"))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group"
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # use label annotation for split groups (shows group labels inside colored blocks)
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group",
        row_annotation = list(.row.split = list(
            type = "label",
            params = list(
                border = FALSE,
                labels_gp = grid::gpar(col = "white", fontsize = 12),
                labels_rot = 0
            )
        ))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # label annotation for column splits
    columns_data <- data.frame(
        columns = paste0("C", 1:10),
        batch = rep(c("A", "B"), each = 5)
    )
    Heatmap(matrix_data, columns_data = columns_data,
        columns_split_by = "batch",
        column_annotation = list(.col.split = list(type = "label"))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # "rownames" annotation for row splits: show the concatenated row names
    # of each split group, wrapped every 3 names (row names and the split
    # title are hidden/shown by default accordingly)
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group",
        row_annotation = list(.row.split = list(
            type = "rownames",
            params = list(sep = ", ", wrap_by = 3)
        ))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
```

```
    # same for column splits with type = "colnames"
    Heatmap(matrix_data, columns_data = columns_data,
        columns_split_by = "batch",
        column_annotation = list(.col.split = list(
            type = "colnames",
            params = list(wrap_by = 3)
        ))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # split title modes: per-slice titles plus the split annotation
    # legend; and "anno" mode using label blocks instead of titles
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group",
        row_title = c("inplace", "legend")
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group",
        row_title = "anno"
    )
}
rownames(matrix_data)[1] <- "R12345"
if (requireNamespace("cluster", quietly = TRUE)) {
    # label annotation for name annotations: show row/column names as colored labels
    Heatmap(matrix_data, rows_data = rows_data,
        row_annotation = list(.row = list(
            type = "label", palette = "Set2", side = "right",
            params = list(labels_rot = 150)
        )),
        column_annotation = list(.col = list(
            type = "label", params = list(labels_rot = 90)
        ))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add labels to the heatmap
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "label",
        base_size = 0.8,
        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",
        base_size = 0.8,
        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)) {
    # Set label color, size, legend and order
    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",
        base_size = 0.6,
        label_name = "Significance",
        label = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if (pv < 0.01)
               list("***", color = "red", size = 12, legend = "p < 0.01", order = 1)
            else if (pv < 0.05)
               list("**", color = "orange", size = 10, legend = "p < 0.05", order = 3)
            else if (pv < 0.1)
               list("*", color = "yellow", size = 8, legend = "p < 0.1", order = 2)
            else NA
        }
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add marks
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "mark",
        mark = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if(pv < 0.01) list("[x]", legend = "p < 0.01")
            else if (pv < 0.02) list("[o]", legend = "p < 0.02")
            else if (pv < 0.03) list("[-]", legend = "p < 0.03")
            else if (pv < 0.05) list("[()]", legend = "p < 0.05")
            else if (pv < 0.06) list("+", legend = "p < 0.06")
            else if (pv < 0.07) list("x", legend = "p < 0.07")
            else if (pv < 0.08) list("[/]", legend = "p < 0.08")
            else if (pv < 0.09) list("[\\]", legend = "p < 0.09")
            else NA
        }
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add labels and marks
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "mark+label",
        label = scales::label_number(accuracy = 0.01),
        mark = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if(pv < 0.01) list("{}", legend = "p < 0.01")
            else if(pv < 0.05) list("[]", legend = "p < 0.05")
```

```
            else NA
        },
        mark_size = 1.5, mark_color = "red"
    )
}
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 name annotations
        # Use .row/.col aliases to disable the built-in name annotations
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_annotation = list(.row = list(params = FALSE)),
        column_annotation = list(.col = list(params = 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 using the .row/.col aliases
    Heatmap(matrix_data,
        column_annotation = list(.col = list(params = list(height = 5))),
        row_annotation = list(.row = list(params = list(width = 5))))
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Per-annotation side control: row name annotation on the right,
    # all other row annotations on the left (.default)
    rows_data2 <- data.frame(
        rows = sample(paste0("R", 1:6), 60, replace = TRUE),
        group = sample(c("X", "Y"), 60, replace = TRUE),
        score = runif(60)
    )
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation = list(
            .default = list(side = "left"),
            .row = list(side = "right"),
            Score = "score"
        ),
        show_row_names = TRUE
    )
}
```

```
if (requireNamespace("cluster", quietly = TRUE)) {
    # Move all row annotations to the right side
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation = list(
            .default = list(side = "right"),
            Score = "score"
        ),
        show_row_names = TRUE
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Split and name annotations on opposite sides:
    # split annotation on the default left, name annotation on the right
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation = list(
            .default = list(side = "left"),
            .row = list(side = "right")
        ),
        show_row_names = TRUE
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Row name label annotation on the right side (text rotated 90° clockwise)
    Heatmap(matrix_data, rows_data = rows_data2,
        row_annotation = list(.row = list(
            type = "label", palette = "Set2", side = "right"
        )),
        show_row_names = TRUE
    )
}

# 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)) {
    # display modes: in-place row names plus the row group annotation
    # legend; and "anno" mode showing the column names as text labels
```

```
    Heatmap(data, rows_by = "r", columns_by = "c", values_by = "value",
        show_row_names = c("inplace", "legend"),
        show_column_names = "anno"
    )
}
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)) {
    # column widths are proportional to the number of bars per column
    # (columns c1 and c2 have 10 and 4 data points)
    bars_data <- data.frame(
        value = rnorm(14),
        r = c(rep("r1", 8), rep("r2", 6)),
        c = c(rep("c1", 6), rep("c2", 2), rep("c1", 4), rep("c2", 2))
    )
    Heatmap(bars_data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "bars")
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # a fraction uses that fraction of each cell's data
    Heatmap(bars_data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "bars", bars_sample = 0.5)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # a whole count samples that many per cell -> equal column widths
    Heatmap(bars_data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "bars", bars_sample = 3)
}
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 <- as.matrix(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 = list(col = "p", type = "ring",
                      params = list(height = grid::unit(10, "mm"), show_legend = FALSE)),
            r2 = list(col = "q", type = "bar"),
            r3 = list(col = "p1", type = "violin",
                       params = list(height = grid::unit(18, "mm")))
        ),
        row_annotation = list(
            .default = list(side = "right"),
            q = list(type = "pie", params = list(width = grid::unit(12, "mm"))),
            p2 = list(type = "density"),
            a = list(type = "simple")
        ),
        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"))
}
```

JitterPlot *Jitter plot*

Description

Draws a jittered point plot showing the distribution of numeric y-values across a discrete x-axis. Each data point is rendered with random jitter along the x-axis (and optionally the y-axis) to reduce overplotting, making it easy to visualise data density, spread, and outliers within each category.

The function supports **x-axis reordering** by y-value summaries (mean or median), **group dodging** via `group_by` to compare subgroups side-by-side, **point labelling** with automatic top-n selection using a configurable distance metric (default: radial distance `y^2 + size^2`), **point highlighting** for emphasis, optional **horizontal reference lines**, and **wide-format input** via `in_form`. Colour control, faceting, and splitting into separate sub-plots via `split_by` are supported.

Usage

```
JitterPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  keep_na = FALSE,
  keep_empty = FALSE,
  sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_text_angle = 0,
  order_by = "-({y}^2 + {size_by}^2)",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  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.
x	A character string specifying the column name for the x-axis. Must be character or factor. Multiple columns can be provided; they are concatenated with x_sep as the separator. When in_form is "wide", the x columns are used as key columns and pivoted to long format (they are not concatenated).
x_sep	A character string used to join multiple x columns. Default "_". Ignored when x is a single column or when in_form is "wide".
y	A character string specifying the numeric column for the y-axis. Required when in_form is "long" (default). When in_form is "wide", y is not required — the values under the x columns are used as y-values.
in_form	A character string specifying the input data format. Either "long" (default) or "wide". In "long" format, x and y are separate columns. In "wide" format, the x columns contain the y-values and are pivoted to a key-value pair.
split_by	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
keep_na	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot. • "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels
sort_x	A character string controlling x-axis level reordering by y-value summaries. One of "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median". "none" leaves the levels as-is. "mean_asc" / "mean" sorts by ascending mean of y. "mean_desc" sorts by descending mean. "median_asc" / "median" sorts by ascending median. "median_desc" sorts by descending median. Default: "none".
flip	A logical value. When TRUE, the x and y axes are swapped via coord_flip and the x-axis factor levels are reversed. Dimension calculation accounts for the flip. Default: FALSE.

group_by — A character vector of column names for dodging the points. Each unique combination becomes a separate dodge group and the points are offset horizontally via position_jitterdodge to reduce overlap. Multiple columns are concatenated with group_by_sep. When NULL (default), no dodging is applied — only jitter via position_jitter.

group_by_sep — A character string used to join multiple group_by columns. Default "_".

group_name — A character string for the dodge-group legend title. When NULL (default), the group_by column name is used.

x_text_angle — A numeric value specifying the angle of the x-axis text.

order_by — A string expression passed to arrange() to determine which points are labelled. Evaluated within each x-group (and facet panel when facet_by is set). Default: "-({y}^2 + {size_by}^2)", which selects points farthest from the origin in y-size radial distance, analogous to VolcanoPlot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value in [0, 1] controlling point transparency. Default: 1.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

shape — A numeric value specifying the point shape (ggplot2 point shape codes). Shapes 21–25 are filled shapes with borders; for these shapes the border behaviour is controlled by border. Default: 21 (filled circle).

border — Controls the border of points when the shape has a border (21–25). If TRUE, the border colour follows the point fill colour (same as the group colour). If a single colour string (e.g. "black"), uses that constant border colour for all points. If FALSE, no border is drawn (NA). Default: "black".

size_by — A numeric column name or a single numeric value controlling point size. When a column name is provided, sizes are scaled using scale_size_area(max_size = 6) and a size legend is shown. When a single numeric value, all points use that constant size. Default: 2.

size_name — A character string for the size legend title. When NULL (default) and size_by is a column, the column name is used. Ignored when size_by is a single numeric value.

size_trans — A function or a function name (as a string) to transform the size_by values for size mapping. The transformed values determine the point size on the plot, but the legend labels show the original (untransformed) values. When NULL (default), no transformation is applied.

y_nbreaks — A numeric value hinting at the number of break intervals for the y-axis. Passed to scale_y_continuous. Default: 4.

jitter_width — A numeric value controlling the amount of horizontal jitter (in x-axis units). Passed to position_jitter / position_jitterdodge. Default: 0.5.

jitter_height — A numeric value controlling the amount of vertical jitter (in y-axis units). Passed to position_jitter / position_jitterdodge. Default: 0.

y_max, y_min — Numeric values or quantile strings (e.g. "q95", "q5") for y-axis limits used in coord_cartesian / coord_flip. When NULL (default), the data range is used. When a quantile string, the corresponding quantile of the y-values is computed via quantile().

y_trans — A character string specifying a transformation for the y-axis (e.g. "log10", "sqrt"). Passed to scale_y_continuous. Default: "identity".

add_bg — A logical value. When TRUE, alternating background stripes are drawn behind the points via bg_layer(), using the x-axis level order. Default: FALSE.

bg_palette — A character string specifying the palette for the background stripe colours. Passed to bg_layer(). Default: "stripe".

bg_palcolor — A character vector of colours for the background stripes. Passed to bg_layer(). When NULL (default), colours are derived from bg_palette.

bg_alpha — A numeric value in [0, 1] for the transparency of the background stripes. Default: 0.2.

add_hline — One or more numeric values specifying y-values at which to draw horizontal reference lines. When NULL (default), no reference lines are drawn.

hline_type — A character string specifying the line type for the horizontal reference line(s). Default: "solid".

hline_width — A numeric value specifying the line width for the horizontal reference line(s). Default: 0.5.

hline_color — A character string specifying the colour for the horizontal reference line(s). Default: "black".

hline_alpha — A numeric value in [0, 1] specifying the alpha (transparency) for the horizontal reference line(s). Default: 1.

labels — A vector of row names or row indices specifying which points to label. When NULL (default) and nlabel > 0, the top nlabel points per x-group are selected automatically.

label_by — A character string naming a column whose values are used as label text. When NULL (default), row names are used as labels.

nlabel — An integer specifying the number of points to label per x-group when labels is NULL. Points are selected by descending order of order_by. Default: 5. Set to 0 to suppress automatic labelling.

label_size, label_fg, label_bg, label_bg_r
: Label aesthetics for geom_text_repel. label_size: text size (default 3). label_fg: text colour (default "black"). label_bg: background (halo) colour for the label text (default "white"). label_bg_r: background border radius (default 0.1).

highlight
: A specification of which points to highlight. Can be: TRUE (highlight all points), a numeric vector of row indices, a single character string parsed as an R expression, or a character vector of row names. When a point is highlighted, an overlay geom_point is drawn on top with highlight_color, highlight_size, and highlight_alpha. Default: NULL (no highlighting).

highlight_color
: A character string specifying the colour of highlighted points. Default: "red2".

highlight_size
: A numeric value specifying the size of highlighted points. Default: 1.

highlight_alpha
: A numeric value in [0, 1] specifying the transparency of highlighted points. Default: 1.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

seed
: A numeric seed for reproducibility. Passed to validate_common_args(). Default: 8525.

combine
: Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow
: Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow
: Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

axes
: A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).

axis_titles
: A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.

guides
: A character string specifying how guides (legends) should be collected across panels (passed to combine_plots()).

design
: A custom layout design for the combined plot (passed to combine_plots()).

...
: Additional arguments.

Value

A ggplot object (when split_by is NULL), a patchwork object (when split_by is provided and combine = TRUE), or a named list of ggplot objects (when combine = FALSE). All ggplot objects have height and width attributes in inches.

split_by workflow

When split_by is provided:

1. validate_common_args() validates the seed.
2. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all relevant columns (x, split_by, group_by, facet_by).
3. The split_by column is validated via check_columns() with force_factor = TRUE. Multiple split_by columns are concatenated with split_by_sep.
4. If split_by is not NULL, the data frame is split (preserving factor level order). If split_by is NULL, the data is wrapped in a single-element list with name ".".
5. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
6. JitterPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
7. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
set.seed(8525)
n <- 180
x <- factor(
    sample(c("A", NA, LETTERS[3:5]), n, replace = TRUE),
    levels = c("A", "B", "C", "D", "E")
)
group <- factor(
    sample(c("G1", NA, "G3"), n, replace = TRUE),
    levels = c("G1", "G2", "G3")
)
size <- rexp(n, rate = 1)
id <- paste0("pt", seq_len(n))
y <- rnorm(n, mean = ifelse(is.na(group), 0, ifelse(group == "G1", 0.5, -0.5))) +
     as.numeric(ifelse(is.na(x), 0, x))/10
df <- data.frame(
  x = x,
  y = y,
  group = group,
  size = size,
  id = id
)

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

```
# Keep empty x levels and NA
JitterPlot(df, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE)

# 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")

# Keep the empty levels only for color coding
# Note the G3 is not blue (which is taken by unused level G2)
JitterPlot(df, x = "x", y = "y", group_by = "group",
    keep_na = TRUE, keep_empty = 'level')

# 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)
```

`LinePlot` *Line plot*

Description

Draws a line plot showing the change of a numeric value across the progression of a categorical x-axis variable. Each x-axis category is rendered as a point connected by a line, with support for multiple grouped series, error bars, highlighted points, background stripes, and a horizontal reference line.

Key features:

- **Colour modes:** lines and points can be coloured by x category (single-series) or by a `group_by` variable (multi-series), or use a single uniform colour.
- **Error bars:** additive error bars via `errorbar_sd`, `errorbar_min`, or `errorbar_max`.
- **Highlighting:** specific points can be emphasised with a different colour and size via indices, row names, or a filter expression.
- **Background stripes:** `add_bg = TRUE` draws alternating bands for visual grouping.
- **Count aggregation:** omit `y` to plot observation counts per x category.

Usage

```
LinePlot(
  data,
  x,
```

```
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  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 = pt_size - 0.75,
  highlight_color = "red2",
  highlight_alpha = 0.8,
  pt_alpha = 1,
  pt_size = 5,
  keep_na = FALSE,
  keep_empty = FALSE,
  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,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  facet_by = NULL,
  facet_scales = "fixed",
  combine = TRUE,
  nrow = NULL,
```

```
  ncol = NULL,
  byrow = TRUE,
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  facet_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
split_by	A character vector of column names to split the data by. Each split level produces a separate sub-plot. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string used to join multiple split_by column values. Default "_".
fill_point_by_x_if_no_group	A logical value. When TRUE (default), points are filled by the x-axis categories via the palette when group_by = NULL. Passed to LinePlotSingle as fill_point_by_x. Has no effect when group_by is set.
color_line_by_x_if_no_group	A logical value. When TRUE (default), lines are coloured by the x-axis categories via the palette when group_by = NULL. Passed to LinePlotSingle as color_line_by_x. Has no effect when group_by is set.
add_bg	A logical value. When TRUE, alternating background stripes are drawn via bg_layer(). Default FALSE.
bg_palette	A character string specifying the palette for the background stripe colours. Default "stripe".
bg_palcolor	A character vector of colours for the background stripes. When NULL (default), colours are derived from bg_palette.

bg_alpha
: A numeric value in [0, 1] for the transparency of background stripes. Default 0.2.

add_errorbars
: A logical value. When TRUE, error bars are added via geom_errorbar(). Requires errorbar_sd or errorbar_min/errorbar_max. Default FALSE.

errorbar_width
: A numeric value for the width of the error bar caps. Default 0.1.

errorbar_alpha
: A numeric value in [0, 1] for the transparency of error bars. Default 1.

errorbar_color
: A character string for the colour of the error bars. When "line", error bars are coloured the same as the lines (by x when color_line_by_x = TRUE, or single colour otherwise). Default "grey30".

errorbar_linewidth
: A numeric value for the line width of error bars. Default 0.75.

errorbar_min
: A character string naming the column with the lower error bar bound. Ignored when errorbar_sd is provided.

errorbar_max
: A character string naming the column with the upper error bar bound. Ignored when errorbar_sd is provided.

errorbar_sd
: A character string naming the column with the standard deviation. When errorbar_min and errorbar_max are not provided, error bars are computed as y +/- errorbar_sd.

highlight
: A vector of row indices, row names, a single string expression (e.g. "y > 10") filtering rows to highlight, or TRUE to highlight all points. When NULL (default), no highlighting is applied.

highlight_size
: A numeric value for the size of highlighted points. Defaults to pt_size - 0.75.

highlight_color
: A character string for the colour of highlighted points. Default "red2".

highlight_alpha
: A numeric value in [0, 1] for the transparency of highlighted points. Default 0.8.

pt_alpha
: A numeric value in [0, 1] for the transparency of points. Default 1.

pt_size
: A numeric value for the point size. Default 5.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

  - FALSE (default): Drop empty factor levels from the data before plotting.
  - TRUE: Keep empty factor levels and show them as a separate category in the plot.

- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

line_type — A character string specifying the line type. Default "solid".

line_width — A numeric value for the line width (in mm). Default 1.

line_alpha — A numeric value in [0, 1] for the transparency of the line. Default 0.8.

add_hline — A numeric value specifying the y-intercept of a horizontal reference line. When FALSE (default), no line is drawn.

hline_type — A character string specifying the line type of the horizontal reference line. Default "solid".

hline_width — A numeric value for the width of the horizontal reference line. Default 0.5.

hline_color — A character string for the colour of the horizontal reference line. Default "black".

hline_alpha — A numeric value in [0, 1] for the transparency of the horizontal reference line. Default 1.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

x_text_angle — A numeric value specifying the angle of the x-axis text.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

combine — Logical; when TRUE (default), per-split plots are combined into a single patchwork object. When FALSE, a named list of ggplot objects is returned.

nrow, ncol — Integer number of rows / columns for the combined layout (passed to wrap_plots).

byrow — Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
facet_args	A list of additional arguments passed to facet_plot() for fine-grained control over faceting (e.g. scales, space, labeller).
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
seed	A numeric seed for reproducibility. Passed to validate_common_args(). Default 8525.
axes	A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides should be collected across panels. Passed to combine_plots().
design	A custom layout specification for combined plots (passed to combine_plots()). Overrides nrow/ncol when specified.
...	Additional arguments.

Value

A ggplot object, a patchwork object (when combine = TRUE with split_by), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. **Column validation** – check_columns() resolves split_by with multi-column concatenation.
2. **NA / empty pre-processing** – process_keep_na_empty() handles keep_na / keep_empty for the split column before splitting, then removes the split column from the per-split lists.
3. **Data splitting** – splits data by split_by levels, preserving factor level order. When split_by = NULL, the data is wrapped in a single-element list with name "...".
4. **Per-split palette / colour** – check_palette() and check_palcolor() resolve per-split palette and colour overrides.
5. **Per-split legend** – check_legend() resolves legend.position and legend.direction per split level.
6. **Per-split title** – when title is a function, it receives the default title (the split level name) and can return a custom string; otherwise title %||% split_level is used.

7. **Dispatch** – each split subset is passed to `LinePlotAtomic`.
8. **Combination** – `combine_plots()` assembles the list of plots via `patchwork::wrap_plots`, honouring `nrow`/`ncol`/`byrow`/`design`.

Examples

```
data <- data.frame(
   x = factor(c("A", "B", "C", "D", "A", "B", "C", "D"), levels = LETTERS[1:6]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = c("G1", "G1", "G1", "G1", "G2", "G2", "G2", "G2"),
   facet = c("F1", "F1", "F2", "F2", "F3", "F3", "F4", "F4")
)

# --- Basic usage ---
LinePlot(data, x = "x", y = "y")
LinePlot(data, x = "x", y = "y", highlight = "group == 'G1'",
   fill_point_by_x_if_no_group = FALSE, color_line_by_x_if_no_group = FALSE)

# --- Grouped lines ---
LinePlot(data, x = "x", y = "y", group_by = "group")
LinePlot(data, x = "x", y = "y", group_by = "group",
   add_hline = 10, hline_color = "red")
LinePlot(data, x = "x", y = "y", group_by = "group", add_bg = TRUE,
   highlight = "y > 10")
LinePlot(data, x = "x", y = "y", group_by = "group", facet_by = "facet")
LinePlot(data, x = "x", y = "y", group_by = "group", split_by = "facet")

# --- Per-split styling ---
LinePlot(data, x = "x", y = "y", split_by = "group",
         palcolor = list(G1 = c("red", "blue"), G2 = c("green", "black")))

# --- keep_na and keep_empty ---
data <- data.frame(
   x = factor(c("A", "B", NA, "D", "A", "B", NA, "D"), levels = LETTERS[1:4]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = factor(c("G1", "G1", "G1", NA, NA, "G3", "G3", "G3"),
     levels = c("G1", "G2", "G3")),
   facet = c("F1", "F1", "F2", "F2", "F3", "F3", "F4", "F4")
)

LinePlot(data, x = "x", y = "y", keep_na = TRUE)
LinePlot(data, x = "x", y = "y", keep_empty = TRUE)
LinePlot(data, x = "x", y = "y", keep_empty = 'level')
LinePlot(data, x = "x", y = "y", group_by = "group", keep_na = TRUE)
LinePlot(data, x = "x", y = "y", group_by = "group", keep_empty = TRUE)
LinePlot(data, x = "x", y = "y", group_by = "group",
   keep_empty = list(x = TRUE, group = 'level'))
```

LinkedHeatmap *Linked Heatmap*

Description

Draw two heatmaps side-by-side with spline link lines connecting matching rows across the two heatmaps. This is the public, exported interface for creating linked-heatmap visualisations.

A typical use case is visualising ligand–receptor interactions: the left heatmap shows ligand expression (rows = ligands, columns = cell sources), the right heatmap shows receptor expression (rows = receptors, columns = cell targets), and link curves connect each ligand to its cognate receptor(s).

Usage

```
LinkedHeatmap(
  data,
  values_by,
  values_fill = NA,
  name = NULL,
  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,
  keep_na = FALSE,
  keep_empty = FALSE,
  rows_orderby = NULL,
  columns_orderby = NULL,
  columns_name = NULL,
  columns_split_name = NULL,
  rows_name = NULL,
  rows_split_name = NULL,
  palette = "RdBu",
  palcolor = NULL,
  palreverse = FALSE,
  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 = 1,
label = identity,
label_size = 10,
label_color = "black",
label_name = "label",
mark = identity,
mark_color = "black",
mark_size = 1,
mark_name = "mark",
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",
cluster_columns = NULL,
cluster_rows = NULL,
show_row_names = NULL,
show_column_names = NULL,
border = TRUE,
title = NULL,
title_params = NULL,
column_title = NULL,
row_title = NULL,
na_col = "grey85",
row_names_side = "right",
column_names_side = "bottom",
row_annotation = NULL,
row_annotation_side = NULL,
row_annotation_palette = NULL,
row_annotation_palcolor = NULL,
row_annotation_type = NULL,
row_annotation_params = NULL,
row_annotation_agg = NULL,
column_annotation = NULL,
```

```
  column_annotation_side = NULL,
  column_annotation_palette = NULL,
  column_annotation_palcolor = NULL,
  column_annotation_type = NULL,
  column_annotation_params = NULL,
  column_annotation_agg = NULL,
  link_width_by = NULL,
  link_width_scale = 5,
  link_color = "grey40",
  link_alpha = 0.6,
  flip = FALSE,
  alpha = 1,
  seed = 8525,
  padding = 15,
  base_size = 1,
  aspect.ratio = NULL,
  draw_opts = list(),
  layer_fun_callback = NULL,
 cell_type = c("tile", "bars", "label", "mark", "label+mark", "mark+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 in long format. Each row represents one observation; columns specify row/column membership for both left and right heatmaps as well as the values to encode as color.
values_by	Default column name for heatmap cell values. Used as fallback when `left_values_by` / `right_values_by` are not explicitly provided via `...`.
values_fill	A value used to fill missing cells in the matrix (passed to `HeatmapAtomic`). Default is `NA` (cells with no data are left empty).
name	Default legend title for the colour scale. Used as fallback when `left_name` / `right_name` are not provided via `...`. The suffixes `" (left)"` / `" (right)"` are appended automatically.
split_by	The column(s) to split data by and plot separately. When `combine = TRUE`, the combined plot's data (`p$data`) contains the rows of every split, tagged with the `split_by` column. For ggraph-based plots (e.g. `Network`, `ClustreePlot`), the node layout of every split is exposed with the split column, and the original

link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.

split_by_sep — The separator for multiple split_by columns. See split_by

rows_by — Default column for rows in both heatmaps. Used as fallback for left_rows_by / right_rows_by.

rows_by_sep — Separator for concatenated rows_by columns.

rows_split_by — Optional column name to split the rows of **both** heatmaps into groups (passed as row_split). When provided, row names in the link table are prefixed with the split level to disambiguate rows across splits.

rows_split_by_sep — Separator for concatenated rows_split_by columns.

columns_by — Default column for columns in both heatmaps. Used as fallback for left_columns_by / right_columns_by.

columns_by_sep — Separator for concatenated columns_by columns.

columns_split_by — Default column to split columns into groups. Used as fallback for left_columns_split_by / right_columns_split_by.

columns_split_by_sep — Separator for concatenated columns_split_by columns.

rows_data, columns_data — Optional data frames providing additional row / column metadata for annotations. Passed through to HeatmapAtomic.

keep_na, keep_empty — Passed through to HeatmapAtomic. See common_args for details.

rows_orderby, columns_orderby — Column name to order rows / columns by (disables clustering when set).

columns_name — Display name for the column annotation.

columns_split_name — Display name for the column split annotation.

rows_name — Display name for the row annotation.

rows_split_name — Display name for the row split annotation.

palette — A character string naming a palette (see show_palettes) or a character vector of colours for the main heatmap colour scale. Default "RdBu". Applied to both heatmaps unless overridden per-side via ....

palcolor — A custom colour vector that overrides palette for the main heatmap colour scale. Applied to both heatmaps unless overridden per-side.

palreverse — Logical; if TRUE, reverse the palette direction.

pie_size_name — Legend title for the pie size when cell_type = "pie".

pie_size — A numeric value or function returning the pie radius. When a function, it receives the count of groups in the pie and should return a radius.

pie_values — A function or string (convertible via match.arg) to compute the value represented by each pie slice. Default "length" counts observations per group.

`pie_name` Default name for the pie legend. Used as fallback for `left_pie_name` / `right_pie_name`.

`pie_group_by` Default column(s) for pie grouping. Used as fallback for `left_pie_group_by` / `right_pie_group_by`.

`pie_group_by_sep`
Separator for concatenated `pie_group_by` columns.

`pie_palette`, `pie_palcolor`
Palette and custom colours for pie slice fill colours.

`bars_sample` Fraction of each cell's observations ($0 < x <= 1$; `1` uses all data) or a whole count > 1 sampled per cell when `cell_type = "bars"`. Column widths are proportional to the number of bars drawn per column. Default 1 (all data).

`label` A function to compute text labels when `cell_type = "label"` (or `"label+mark"`). Receives the aggregated value for a cell and optionally row/column indices and names. See `HeatmapAtomic` for the full dispatch contract.

`label_size` Default point size for label text (used as fallback when the `label` function does not return a `size` field).

`label_color` Default colour for label text (used as fallback when the `label` function does not return a `color` field).

`label_name` Legend title for the label colour scale.

`mark` A function to compute mark symbols when `cell_type = "mark"` (or `"label+mark"`). Same dispatch contract as `label`. See `HeatmapAtomic` for supported mark types.

`mark_color` Default mark colour (fallback).

`mark_size` Default mark stroke width in pt (fallback).

`mark_name` Legend title for the mark colour scale.

`violin_fill` A character vector of colours to use as fill for violin plots when `cell_type = "violin"`. If `NULL`, the annotation colour is used.

`boxplot_fill` A character vector of colours to use as fill for boxplots when `cell_type = "boxplot"`. If `NULL`, the annotation colour is used.

`dot_size` Dot size when `cell_type = "dot"`. Can be a numeric value or a function.

`dot_size_name` Legend title for the dot size.

`legend_items` A named numeric vector specifying custom legend entries for the main colour scale. Names become the displayed labels.

`legend_discrete`
Logical; if `TRUE`, treat the main colour scale as discrete.

`legend.position`
A character string specifying where to place the combined legend: `"right"` (default), `"left"`, `"top"`, `"bottom"`, or `"none"`.

`legend.direction`
Legend stacking direction: `"vertical"` (default) or `"horizontal"`.

`lower_quantile`, `upper_quantile`
Quantiles used for clipping the colour scale when `lower_cutoff` / `upper_cutoff` are `NULL`. Defaults are 0 and 0.99 respectively.

lower_cutoff, upper_cutoff
: Explicit cutoffs for the colour scale. Values outside the range are clamped (winsorized). Override lower_quantile / upper_quantile when set.

add_bg
: Logical; if TRUE, add a background fill behind non-tile cell types. Not used for cell_type = "tile" or "bars".

bg_alpha
: Numeric in $[0, 1]$ for background transparency.

add_reticle
: Logical; if TRUE, draw a reticle (crosshair pattern) over the heatmap.

reticle_color
: Colour for the reticle lines.

cluster_columns
: Logical; cluster columns in both heatmaps. NULL lets HeatmapAtomic decide.

cluster_rows
: Default clustering setting for rows. Used as fallback for left_cluster_rows / right_cluster_rows.

show_row_names, show_column_names
: Logical; show row/column names. May also be a character vector of display modes; see Heatmap.

border
: Logical; draw a border around each heatmap. Default TRUE.

title
: A character string for the overall plot title. A function can be used to generate a dynamic title from the default. Note that, left_title and right_title are used to set the title for each heatmap, and title is used to set the overall title for the combined plot.

title_params
: A list of parameters passed to grid::grid.text() to control the title appearance. Default is list(gp = gpar(fontsize = 14, fontface = "bold")).

column_title, row_title
: Character title displayed above the columns / beside the rows of each heatmap. May also be a character vector of display modes; see Heatmap.

na_col
: Colour used for NA cells. Default "grey85".

row_names_side
: Default side for row names. Used as fallback for left_row_names_side / right_row_names_side. Default "right".

column_names_side
: Side for column names. Default "bottom".

row_annotation
: A structured list specifying row annotations. See HeatmapAtomic for the full specification.

row_annotation_side
: **Deprecated**: use row_annotation with the side sub-key instead. Used as fallback for left_row_annotation_side / right_row_annotation_side. Default "left".

row_annotation_palette
: **Deprecated**: use row_annotation with the palette sub-key instead.

row_annotation_palcolor
: **Deprecated**: use row_annotation with the palcolor sub-key instead.

row_annotation_type
: **Deprecated**: use row_annotation with the type sub-key instead.

row_annotation_params
: **Deprecated**: use row_annotation with the params sub-key instead.

`row_annotation_agg`
: **Deprecated**: use `row_annotation` with the `agg` sub-key instead.

`column_annotation`
: A structured list specifying column annotations. See `HeatmapAtomic` for the full specification.

`column_annotation_side`
: **Deprecated**: use `column_annotation` with the `side` sub-key instead.

`column_annotation_palette`
: **Deprecated**: use `column_annotation` with the `palette` sub-key instead.

`column_annotation_palcolor`
: **Deprecated**: use `column_annotation` with the `palcolor` sub-key instead.

`column_annotation_type`
: **Deprecated**: use `column_annotation` with the `type` sub-key instead.

`column_annotation_params`
: **Deprecated**: use `column_annotation` with the `params` sub-key instead.

`column_annotation_agg`
: **Deprecated**: use `column_annotation` with the `agg` sub-key instead.

`link_width_by`
: Optional column name in `data` whose values determine the stroke width of each link line (e.g. interaction strength). Values are min-max scaled to $[0, 1]$ and multiplied by `link_width_scale`. You can also pass a numeric value to use a constant width for all links.

`link_width_scale`
: Numeric scaling factor applied to the normalised link intensity values to produce final line widths (`lwd`). Default 5.

`link_color`
: Colour of the link spline curves. Default `"grey30"`.

`link_alpha`
: Alpha transparency of link curves in $[0, 1]$. Default 0.8.

`flip`
: Logical; must be `FALSE` for linked heatmaps (flipping is not supported). Default `FALSE`.

`alpha`
: Alpha transparency for heatmap cells in $[0, 1]$.

`seed`
: Random seed for reproducibility. Default 8525.

`padding`
: Padding around the heatmap in CSS order (top, right, bottom, left). Supports 1–4 values. Default 15 (mm).

`base_size`
: A positive numeric scalar used as a scaling factor for the overall heatmap size. Default 1 (no scaling). Values > 1 enlarge all cell dimensions proportionally.

`aspect.ratio`
: Height-to-width ratio of a single heatmap cell. When `NULL` (default), sensible defaults are chosen per `cell_type` (e.g. 1 for tiles, 0.5 for bars, 2 for violins).

`draw_opts`
: A named list of additional arguments passed to `draw,HeatmapList-method`. Internally managed arguments (`padding`, `show_heatmap_legend`, etc.) take precedence.

`layer_fun_callback`
: A function to add custom graphical layers on top of each heatmap cell. Receives `j`, `i`, `x`, `y`, `w`, `h`, `fill`, `sr`, `sc`. See `Heatmap` for details.

`cell_type`
: The type of cell to render. One of `"tile"` (default), `"bars"`, `"label"`, `"mark"`, `"label+mark"` (or `"mark+label"`), `"dot"`, `"violin"`, `"boxplot"`, `"pie"`. Different cell types use different `cell_fun` / `layer_fun` implementations.

`cell_agg` A function to aggregate values within each cell when `cell_type = "tile"` or `"label"`. Default is `mean`.

`combine` Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow` A numeric value specifying the number of rows in the facet.

`ncol` A numeric value specifying the number of columns in the facet.

`byrow` A logical value indicating whether to fill the plots by row.

`axes` A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments passed to `LinkedHeatmapAtomic`. All parameters listed above (and those inherited from `LinkedHeatmapAtomic`) can be specified with `left_` or `right_` prefixes for per-side control (e.g. `left_palette = "Blues"`, `right_palette = "Reds"`). Unprefixed arguments apply to both sides. Also forwarded to `Heatmap` after prefix stripping.

Value

A `patchwork` object (class `wrap_plots`) with `height` and `width` attributes (in inches). When `combine = FALSE`, a named list of such objects, one per `split_by` level.

Left / right specification

Parameters that differ between the two heatmaps are prefixed with `left_` or `right_`. Shared parameters (e.g. `palette`, `cell_type`, `cluster_columns`) apply to both sides but can be overridden per-side via the `...` argument. Every parameter listed below can also be passed with a `left_` or `right_` prefix in `...` for full per-side control.

The `...` argument is also forwarded to `Heatmap` after prefix-stripping, allowing direct access to `ComplexHeatmap` parameters (e.g. `left_row_names_gp`, `right_column_names_rot`).

Split-by support

When `split_by` is provided, the data is partitioned into subsets and an independent linked-heatmap pair is produced for each level. The results are combined via `wrap_plots` according to `nrow`, `ncol`, `byrow`, and `design`. Per-split `palette`, `palcolor`, and `legend.position` can be specified as named lists keyed by split level.

Dimension calculation

Cell dimensions are pre-computed from `cell_type`, `aspect.ratio`, `base_size`, and the unique row/column counts in the data (accounting for split groups). These exact dimensions are passed to `ComplexHeatmap::Heatmap` so cells have guaranteed physical sizes, ensuring that the two heatmaps' bodies align precisely and that link line endpoints land on the correct rows. The final `height` / `width` attributes on the returned object include legend space and are clamped to $[4, 64]$ inches with aspect-ratio correction.

See Also

`Heatmap`

Examples

```
set.seed(8525)
# Define sparse ligand-receptor pairs
pairs_df <- data.frame(
    ligand = c("Ligand1", "Ligand2", "Ligand3", "Ligand4", "Ligand5",
               "Ligand1", "Ligand3", "Ligand5"),
    receptor = c("Receptor1", "Receptor2", "Receptor1", "Receptor3", "Receptor4",
                 "Receptor5", "Receptor2", "Receptor5"),
    stringsAsFactors = FALSE
)
sources <- paste0("Source", 1:4)
targets <- paste0("Target", 1:6)

# Expand pairs across all sources and targets
data <- merge(
    merge(pairs_df, data.frame(source = sources, stringsAsFactors = FALSE)),
    data.frame(target = targets, stringsAsFactors = FALSE)
)
data$split <- sample(c("A", "B"), nrow(data), replace = TRUE)
data$ligand_expr <- runif(nrow(data), 0, 10)
data$receptor_expr <- runif(nrow(data), 0, 10)
```

```
data$intensity <- runif(nrow(data), 0, 1)

if (requireNamespace("ComplexHeatmap", quietly = TRUE)) {
    LinkedHeatmap(
        data,
        show_column_names = TRUE,
        column_names_side = "bottom",
        show_row_names = FALSE,
        left_row_names_side = "right",
        left_rows_by = "ligand",
        left_columns_by = "source",
        left_values_by = "ligand_expr",
        left_name = "Ligand",
        left_row_annotation = list(
            .row = list(
                type = "label",
                params = list(labels_rot = 0),
                side = "right"
            )
        ),
        right_cluster_rows = FALSE,
        right_row_names_side = "left",
        right_row_annotation = list(
            .row = list(
                type = "label",
                params = list(labels_rot = 0),
                side = "left"
            )
        ),
        right_rows_by = "receptor",
        right_columns_by = "target",
        right_values_by = "receptor_expr",
        right_name = "Receptor",
        link_width_by = "intensity"
    )
}
```

`ManhattanPlot` *Manhattan plot*

Description

Renders a publication-quality Manhattan plot for genetic association results. The y-axis displays $-\log_{10}(p)$ (or a user-specified transformation) of p-values, and the x-axis shows genomic positions organised by chromosome. Each chromosome is rendered in alternating colours, and configurable horizontal dashed lines mark genome-wide significance thresholds.

The function is adapted from `ggmanh::manhattan_plot()` with extended control over point appearance, variant labels, highlighting, data thinning, y-axis rescaling, and `split_by` support for creating multi-panel layouts (e.g. faceted by cohort or phenotype).

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_this",
  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 = axes,
  guides = NULL,
  facet_by = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
chr_by	A character string specifying the column name for chromosome identifiers. Default: "chr".
pos_by	A character string specifying the column name for genomic positions (integer or numeric). Default: "pos".
pval_by	A character string specifying the column name for p-values (numeric). Default: "pval".
split_by	The column(s) to split data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string used to concatenate multiple split_by column values. Default: "_".
label_by	A character string specifying the column name for variant labels. Only variants with non-empty values in this column will be labelled. Default: NULL (no labels).
chromosomes	A character or numeric vector specifying which chromosomes to include and/or their display order. When NULL (the default), all chromosomes present in the data are plotted in their natural factor order. A single value filters to that chromosome; a vector reorders and subsets.
pt_size	A numeric value specifying the size of the points. Default: 0.75.
pt_color	A character string specifying a single colour for all background (non-highlighted) points. When NULL (the default), alternating chromosome colours from palette / palcolor are used. Typically set to "grey80" when highlight is used with a distinct highlight_color.
pt_alpha	A numeric value in [0, 1] specifying the transparency of the points. Default: alpha (aliased parameter).
pt_shape	A numeric value specifying the shape of the points. Default: 19 (filled circle).
label_size	A numeric value specifying the font size of the variant labels. Default: 3.
label_fg	A character string specifying the colour of the variant labels. When NULL (the default), each label inherits the colour of its corresponding point.
highlight	Either a numeric vector of row indices or a character string containing an R expression (parsed via rlang::parse_expr()) to select variants to highlight. Default: NULL (no highlighting).

highlight_color
: A character string specifying the colour of highlighted points. When NULL (the default), highlighted points inherit the chromosome colour from the underlying geom_point() layer.

highlight_size
: A numeric value specifying the size of highlighted points. Default: 1.5.

highlight_alpha
: A numeric value in [0, 1] specifying the transparency of highlighted points. Default: 1.

highlight_shape
: A numeric value specifying the shape of highlighted points. Default: 19 (filled circle).

preserve_position
: A logical value. When TRUE (the default), the width of each chromosome segment reflects its number of variants and variant positions are correctly scaled. When FALSE, all chromosomes have equal width and variants are equally spaced.

chr_gap_scaling
: A numeric scaling factor for the gap between chromosomes. Larger values increase the gap. Default: 1.

pval_transform
: A function or character string that can be evaluated to a function for transforming p-values. Default: "-log10", which computes $-\log_{10}(p)$. Other examples: "-log2" or a custom function(x) -log10(x).

signif
: A numeric vector of significance thresholds to draw as horizontal dashed lines. Default: c(5e-8, 1e-5).

signif_color
: A character vector of colours for the significance threshold lines, of equal length as signif. When NULL (the default), the smallest threshold is coloured black and the rest grey.

signif_rel_pos
: A numeric value between 0.1 and 0.9 specifying the relative position of the y-axis jump when rescaling is active. Default: 0.2.

signif_label
: A logical value. When TRUE (the default), significance threshold values are annotated on the plot.

signif_label_size
: A numeric value for the font size of the significance threshold labels. Default: 3.5.

signif_label_pos
: A character string specifying where to place the significance threshold labels: "left" (default) or "right".

thin
: A logical value indicating whether to thin dense data by sampling points per horizontal partition. Defaults to TRUE when chromosomes selects fewer chromosomes than in the data, and FALSE otherwise.

thin_n
: An integer specifying the maximum number of points per horizontal partition after thinning. Default: 1000.

thin_bins
: An integer specifying the number of horizontal bins for thinning. Default: 200.

rescale
: A logical value. When TRUE (the default), the y-axis is automatically rescaled (broken axis) if extreme significance values would otherwise compress the main data cloud.

rescale_ratio_threshold
: A numeric threshold for triggering y-axis rescaling. The ratio is computed as `ceiling(max(log10pval) / 5) * 5 / signif_jump`. Default: 5.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

alpha
: A numeric value specifying the transparency of the plot.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

seed
: A numeric seed for reproducibility. Passed to `validate_common_args()`. Default: 8525.

combine
: A logical value. When TRUE (the default), the list of per-split plots is combined into a single patchwork object. When FALSE, returns the raw list.

nrow, ncol, byrow
: Integers controlling the layout of combined plots via `patchwork::wrap_plots()`. `byrow = TRUE` fills the layout row-wise.

axes, axis_titles
: Strings controlling how axes and axis titles are handled across combined plots. Passed to `combine_plots()`. See ?patchwork::wrap_plots for options ("keep", "collect", "collect_x", "collect_y").

guides
: A string controlling guide collection across combined plots. Passed to `combine_plots()`.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

design
: A custom layout specification for combined plots. Passed to `combine_plots()`. When specified, nrow, ncol, and byrow are ignored.

...
: Additional arguments.

Value

A ggplot object (single plot, no split_by), a patchwork object (when combine = TRUE with split_by), or a named list of ggplot objects (when combine = FALSE). Each individual plot carries height and width attributes.

split_by Workflow

When split_by is provided:

1. **Column validation** — check_columns() resolves split_by with force_factor = TRUE, allow_multi = TRUE, and concat_multi = TRUE. For GRanges inputs, validation is performed on the @elementMetadata slot.
2. **GRanges support** — data can be a data.frame or a GenomicRanges::GRanges object. When GRanges is used, split_by is read from the metadata columns.
3. **Data splitting** — drops unused split_by levels, splits data by split_by (preserving factor level order), and wraps into a named list. When split_by is NULL, the data is wrapped as a single-element list with name "...".
4. **Per-split palette / colour** — check_palette() and check_palcolor() resolve per-split palette and colour overrides.
5. **Per-split title** — when title is a function, it receives the default title (the split level name) and can return a custom string; otherwise title %||% split_level is used.
6. **Dispatch** — each split subset is passed to ManhattanPlotAtomic.
7. **Combination** — combine_plots() assembles the list of plots via patchwork::wrap_plots, honouring nrow / ncol / byrow / design.

Note

facet_by is not supported by this plot type and triggers a warning if provided. Use split_by instead to produce comparable multi-panel layouts.

Examples

```
set.seed(1000)

nsim <- 50000

# --- Data simulation ---
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"))
options(repr.plot.width=10, repr.plot.height=5)

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

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

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

# --- Customized p-value transformation and significance threshold line colors ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
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"))
}

# --- Different palette and no significance threshold labels ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   palette = "Set1", signif_label = FALSE)
}

# --- Reverse palette and label position on the right ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   palette = "Set1", palreverse = TRUE, signif_label_pos = "right")
}

# --- Single chromosome ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated P.Values", chromosomes = 5)
}

# --- Chromosome subset and reorder ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated P.Values", chromosomes = c(20, 4, 6))
}

tmpdata <- data.frame(
  "chromosome" = c(rep(5, 10), rep(21, 5)),
  "position" = c(sample(250000:250100, 10, replace = FALSE),
    sample(590000:600000, 5, replace = FALSE)),
  "P.value" = c(10^-(rnorm(10, 100, 3)), 10^-rnorm(5, 9, 1)),
  "cohort" = c(rep("A", 10), rep("B", 5))
)

simdata <- rbind(simdata, tmpdata)
simdata$chromosome <- factor(simdata$chromosome, c(1:22, "X"))

# --- Disable y-axis rescaling ---
```

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

# --- Y-axis rescaling with custom break position ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
    simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated P.Values - Significant", rescale = TRUE, signif_rel_pos = 0.5)
}

sig <- simdata$P.value < 5e-07

simdata$label <- ""
simdata$label[sig] <- sprintf("Label: %i", 1:sum(sig))
simdata$label2 <- ""
i <- (simdata$chromosome == 5) & (simdata$P.value < 5e-8)
simdata$label2[i] <- paste("Chromosome 5 label", 1:sum(i))

# --- Variant labels ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(simdata, label_by = "label", pval_by = "P.value", chr_by = "chromosome",
    pos_by = "position", title = "Simulated P.Values with labels", label_size = 4)
}

# --- Variant labels with custom color ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(simdata, label_by = "label2", pval_by = "P.value", chr_by = "chromosome",
    pos_by = "position", title = "Simulated P.Values with labels",
    label_size = 3, label_fg = "black")
}

simdata$color <- "Not Significant"
simdata$color[simdata$P.value <= 5e-8] <- "Significant"

# --- Highlight points with custom shape ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(simdata, title = "Highlight Points with shapes",
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    highlight = "color == 'Significant'", highlight_color = NULL, highlight_shape = 6,
    highlight_size = 5, pt_alpha = 0.2, pt_size = 1)
}

# --- Highlight points with custom color ---
if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(simdata, title = "Highlight Points",
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    highlight = "color == 'Significant'", highlight_color = "black",
    pt_color = "lightblue", pt_alpha = 0.2, pt_size = 0.1)
}
```

Network *Network*

Description

Draws a network graph from a links (edge list) data frame and an optional nodes (vertex metadata) data frame. The graph is constructed via igraph, laid out with igraph layout algorithms, and rendered with ggraph. Supports directed or undirected edges, variable link widths/linetypes/colours, node sizes/shapes/colours/fills, community detection with enclosure marks, automatic node labels, and a wide range of layout options.

When `links` (and optionally `nodes`) contain a `split_by` column, separate sub-plots are generated for each split level and combined via patchwork. Unlike most other plot types, `Network` operates on two data frames; splitting may affect both.

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,
  palreverse = FALSE,
  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_this",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

links	A data frame containing the edge list. Must contain the `from` and `to` columns specifying source and target node identifiers. Additional columns can be referenced by other parameters (e.g., `link_weight_by`, `link_type_by`, `link_color_by`).

nodes — An optional data frame of node metadata. When provided, columns such as node_size_by, node_color_by, node_shape_by, and node_fill_by can reference its columns. When NULL, the node set is inferred from the unique values in the from and to columns. If a single character string starting with "@", the nodes data frame is extracted from the corresponding attribute of links (e.g. "@nodes" extracts attr(links, "nodes")).

split_by — The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.

split_by_sep — The separator for multiple split_by columns. See split_by

split_nodes — A logical value. When TRUE and split_by is provided, the nodes data frame is split by the same split_by column in addition to the links. Both data frames must have a column with the same name as split_by. Default FALSE.

from — A character string specifying the column name in links for the source node identifiers. Defaults to "from", or the first column of links if that column name does not exist. Multiple columns can be provided; they are concatenated with from_sep.

from_sep — A character string to join multiple from columns. Default "_". Ignored when from is a single column.

to — A character string specifying the column name in links for the target node identifiers. Defaults to "to", or the second column of links if that column name does not exist. Multiple columns can be provided; they are concatenated with to_sep.

to_sep — A character string to join multiple to columns. Default "_". Ignored when to is a single column.

node_by — A character string specifying the column name in nodes for the node identifiers. These must match the values in the from / to columns of links. Defaults to "name", or the first column of nodes if that column name does not exist. Multiple columns can be provided; they are concatenated with node_by_sep.

node_by_sep — A character string to join multiple node_by columns. Default "_". Ignored when node_by is a single column.

link_weight_by — A numeric value or a character string. If numeric, all edges receive that constant line width. If a column name, the edge line width is mapped to that column. Default 2.

link_weight_name — A character string for the link weight legend title. When NULL (default), the column name from link_weight_by is used. Only relevant when link_weight_by is a column name.

link_type_by — A character string or a column name specifying the edge linetype. Can be "solid", "dashed", "dotted", etc. If a column name from links is supplied, the linetype is mapped to that column (with a version check for ggplot2 4.0.0, where mapping is unsupported and a warning is issued). Default "solid".

link_type_name A character string for the link linetype legend title. When NULL (default), the column name from link_type_by is used. Only relevant when link_type_by is a column name.

node_size_by A numeric value or a character string. If numeric, all nodes receive that constant point size. If a column name, the size is mapped to that column. Default 15.

node_size_name A character string for the node size legend title. When NULL (default), the column name from node_size_by is used. Only relevant when node_size_by is a column name.

node_color_by A character string specifying the node colour. If a colour name or hex code (e.g. "black"), all nodes receive that constant colour. If a column name from nodes is supplied, the colour is mapped to that column. Default "black".

node_color_name
A character string for the node colour legend title. When NULL (default), the column name from node_color_by is used. Only relevant when node_color_by is a column name.

node_shape_by A numeric value or a character string. If numeric, all nodes receive that constant shape (see shape). If a column name, the shape is mapped to that column (cast to factor). Default 21 (filled circle with border).

node_shape_name
A character string for the node shape legend title. When NULL (default), the column name from node_shape_by is used. Only relevant when node_shape_by is a column name.

node_fill_by A character string specifying the node fill colour. If a colour name or hex code (e.g. "grey20"), all nodes receive that constant fill. If a column name from nodes is supplied, the fill is mapped to that column. Default "grey20".

node_fill_name A character string for the node fill legend title. When NULL (default), the column name from node_fill_by is used. Only relevant when node_fill_by is a column name.

link_alpha A numeric value specifying the transparency (alpha) of the edge lines. Between 0 (invisible) and 1 (opaque). Default 1.

node_alpha A numeric value specifying the fill transparency of the nodes. Only applies when node_shape_by is one of the filled shapes (21–25). Default 0.95.

node_stroke A numeric value specifying the border stroke width of the node points. Default 1.5.

cluster_scale A character string specifying which node aesthetic is overridden by cluster membership. One of "fill", "color", or "shape". The value is matched via match.arg; default is "fill".

node_size_range
A numeric vector of length 2 giving the minimum and maximum node size (in ggplot2 point units) when node_size_by is a column name. Default c(5, 20).

link_weight_range
A numeric vector of length 2 giving the minimum and maximum edge line width (in mm) when link_weight_by is a column name. Default c(0.5, 5).

link_arrow_offset
: A numeric value (in points) specifying the offset distance for the arrow end cap from the target node. Prevents arrow heads from overlapping the node points. Only relevant when `directed = TRUE`. Default `20`.

link_curvature
: A numeric value controlling the curvature of the edges. `0` (default) produces straight edges; positive values curve them away from the direct path.

link_color_by
: A character string controlling how edge colour is determined. Options:
  - `"from"` (default) – colour follows the source node's fill or colour aesthetic.
  - `"to"` – colour follows the target node's fill or colour.
  - A column name from `links` – colour is mapped directly to that column.

link_color_name
: A character string for the edge colour legend title. Only used when `link_color_by` is a column name (not `"from"` or `"to"`). When `NULL` (default), the column name is used.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

link_palette
: A character string specifying the palette for edge colours when they are mapped. When `link_color_by` is `"from"` or `"to"`, defaults to the node `palette`. Otherwise defaults to `"Set1"`.

link_palcolor
: A character vector specifying custom colours for the edge palette. When `link_color_by` is `"from"` or `"to"`, defaults to the node `palcolor`. Otherwise defaults to `NULL`.

directed
: A logical value. When `TRUE`, edges are drawn with arrow heads and an end-cap offset. Default `TRUE`.

layout
: A character string or an `igraph_layout_spec` object specifying the node placement algorithm. Built-in shortcuts: `"circle"` (circular layout), `"tree"` (hierarchical tree), `"grid"` (grid layout). Any other string is prefixed with `"layout_with_"` and called as an igraph function (e.g. `"fr"` for Fruchterman–Reingold, `"kk"` for Kamada–Kawai). Default `"circle"`.

cluster
: A character string specifying the community detection algorithm. One of `"none"`, `"fast_greedy"`, `"walktrap"`, `"edge_betweenness"`, `"infomap"`, or a custom clustering function from igraph. When not `"none"`, cluster membership overrides the aesthetic selected by `cluster_scale`. Default `"none"`.

add_mark
: A logical value. When `TRUE` (and `cluster != "none"`), an enclosure mark is drawn around each cluster's nodes. Default `FALSE`.

mark_expand
: A `unit` object specifying the extra space around points within a cluster mark. Default `unit(10, "mm")`.

mark_type
: A character string specifying the mark geometry. One of `"hull"`, `"ellipse"`, `"rect"`, or `"circle"`, corresponding to ggforce's `geom_mark_hull`, `geom_mark_ellipse`, `geom_mark_rect`, and `geom_mark_circle`. The value is matched via `match.arg`; default is `"hull"`.

mark_alpha
: A numeric value for the fill transparency of cluster marks. Default 0.1.

mark_linetype
: A numeric or character value specifying the border line type of the cluster marks. Default 1 (solid).

add_label
: A logical value. When TRUE (default), node identifiers are drawn as repulsive text labels via geom_text_repel.

label_size
: A numeric value for the font size of node labels. Scaled by the theme base size. Default 3.

label_fg
: A character string specifying the text colour of node labels. Default "white".

label_bg
: A character string specifying the background colour of node labels. Default "black".

label_bg_r
: A numeric value specifying the background box radius (as a fraction of label height). Passed to geom_text_repel's bg.r argument. Default 0.1.

arrow
: A arrow object for the link arrow heads. Only used when directed = TRUE. Default is arrow(type = "closed", length = unit(0.1, "inches")).

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A `ggplot` object (no `split_by`), a `patchwork` object (`combine = TRUE`), or a named list of `ggplot` objects (`combine = FALSE`), each with `height` and `width` attributes in inches.

split_by workflow

When `split_by` is provided:

1. **Column validation** – The `split_by` column is validated in `links` via `check_columns`, force-converted to a factor, and empty levels are dropped.
2. **Node split** – If `split_nodes = TRUE` and `nodes` is provided, the same `split_by` column is validated in `nodes`. It must be identical in name to the links `split_by` or an error is raised. Empty levels are also dropped.
3. **Data splitting** – The `links` data frame is split by the `split_by` levels into a named list, preserving factor level order.
4. **Attach node splits** – If `split_nodes = TRUE`, the `nodes` data frame is split identically. Each split's node data is attached as the `"nodes"` attribute on the corresponding links split.
5. **Dispatch to atomic** – `NetworkAtomic` is called for each split. The `nodes` argument is passed as `"@nodes"` when `split_nodes = TRUE` so that it is extracted from the attribute. If `title` is a function, it receives the split level name for dynamic title generation.
6. **Combination** – Results are combined via `combine_plots()` (when `combine = TRUE`) or returned as a named list of `ggplot` objects.

7. **Combined data** – When `combine = TRUE`, the combined plot's `data` (`p$data`) exposes the node layout of every split, each row tagged with the `split_by` level. The `"graph"` attribute carries a combined graph of all splits' nodes and links, and the original link rows (tagged with the `split_by` column) are available in the `"edges"` attribute.

Examples

```
# Create example data
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)
)

# Basic network
Network(relations, actors)

# Blank theme with no coordinate axes
Network(relations, actors, theme = "theme_blank",
        theme_args = list(add_coord = FALSE))

# Mapped aesthetics with custom layout
Network(relations, actors,
        link_weight_by = "friendship",
        node_size_by = "age",
        link_weight_name = "FRIENDSHIP",
        node_fill_by = "gender",
        link_color_by = "to",
        link_type_by = "type",
        node_color_by = "black",
        layout = "circle",
        link_curvature = 0.2)

# Tree layout with clustering and marks
Network(relations, actors, layout = "tree",
        directed = FALSE, cluster = "fast_greedy",
        add_mark = TRUE)

# Split by a column
Network(relations, actors, split_by = "type")
```

palette_list *A list of palettes for use in data visualization*

Description

A list of palettes for use in data visualization

Examples

```
## Not run:
if (interactive()) {
  library(stringr)
  library(RColorBrewer)
  library(Redmonder)
  library(rcartocolor)
  library(nord)
  library(viridis)
  library(pals)
  library(dichromat)
  library(jcolors)
  library(scales)
  library(ggthemes)
  syspals <- utils::getFromNamespace("syspals", "pals")
  brewer.pal.info <- RColorBrewer::brewer.pal.info
  ggsci_db <- utils::getFromNamespace("ggsci_db", "ggsci")
  redmonder.pal.info <- Redmonder::redmonder.pal.info
  metacartocolors <- rcartocolor::metacartocolors
  rownames(metacartocolors) <- metacartocolors$Name
  nord_palettes <- nord::nord_palettes
  viridis_names <- c("magma", "inferno", "plasma", "viridis", "cividis", "rocket",
     "mako", "turbo")
  viridis_palettes <- lapply(stats::setNames(viridis_names, viridis_names),
     function(x) viridis::viridis(100, option = x))
  ocean_names <- names(syspals)[grep("ocean", names(syspals))]
  ocean_palettes <- syspals[ocean_names]
  dichromat_palettes <- dichromat::colorschemes
  jcolors_names <- paste0("jcolors-", c("default", "pal2", "pal3", "pal4", "pal5",
     "pal6", "pal7", "pal8", "pal9", "pal10", "pal11", "pal12", "rainbow"))
  custom_names <- c("jet", "simspec", "GdRd")
  custom_palettes <- list(
    oompaBase::jetColors(N = 100),
    c("#c22b86", "#f769a1", "#fcc5c1", "#253777", "#1d92c0", "#9ec9e1", "#015b33",
        "#42aa5e", "#d9f0a2", "#E66F00", "#f18c28", "#FFBB61"),
    c("gold", "red3")
  )
  names(custom_palettes) <- custom_names
  seurat_discrete_palettes <- list(
     alphabet = c(
       "#F0A0FF", "#0075DC", "#993F00", "#4C005C", "#191919", "#005C31",
       "#2BCE48", "#FFCC99", "#808080", "#94FFB5", "#8F7C00", "#9DCC00",
```

```
    "#C20088", "#003380", "#FFA405", "#FFA8BB", "#426600", "#FF0010",
    "#5EF1F2", "#00998F", "#E0FF66", "#740AFF", "#990000", "#FFFF80",
    "#FFE100", "#FF5005"
  ),
  alphabet2 = c(
    "#AA0DFE", "#3283FE", "#85660D", "#782AB6", "#565656", "#1C8356",
    "#16FF32", "#F7E1A0", "#E2E2E2", "#1CBE4F", "#C4451C", "#DEA0FD",
    "#FE00FA", "#325A9B", "#FEAF16", "#F8A19F", "#90AD1C", "#F6222E",
    "#1CFFCE", "#2ED9FF", "#B10DA1", "#C075A6", "#FC1CBF", "#B00068",
    "#FBE426", "#FA0087"
  ),
  glasbey = c(
    "#0000FF", "#FF0000", "#00FF00", "#000033", "#FF00B6", "#005300",
    "#FFD300", "#009FFF", "#9A4D42", "#00FFBE", "#783FC1", "#1F9698",
    "#FFACFD", "#B1CC71", "#F1085C", "#FE8F42", "#DD00FF", "#201A01",
    "#720055", "#766C95", "#02AD24", "#C8FF00", "#886C00", "#FFB79F",
    "#858567", "#A10300", "#14F9FF", "#00479E", "#DC5E93", "#93D4FF",
    "#004CFF", "#F2F318"
  ),
  polychrome = c(
    "#5A5156", "#E4E1E3", "#F6222E", "#FE00FA", "#16FF32", "#3283FE",
    "#FEAF16", "#B00068", "#1CFFCE", "#90AD1C", "#2ED9FF", "#DEA0FD",
    "#AA0DFE", "#F8A19F", "#325A9B", "#C4451C", "#1C8356", "#85660D",
    "#B10DA1", "#FBE426", "#1CBE4F", "#FA0087", "#FC1CBF", "#F7E1A0",
    "#C075A6", "#782AB6", "#AAF400", "#BDCDFF", "#822E1C", "#B5EFB5",
    "#7ED7D1", "#1C7F93", "#D85FF7", "#683B79", "#66B0FF", "#3B00FB"
  ),
  stepped = c(
    "#990F26", "#B33E52", "#CC7A88", "#E6B8BF", "#99600F", "#B3823E",
    "#CCAA7A", "#E6D2B8", "#54990F", "#78B33E", "#A3CC7A", "#CFE6B8",
    "#0F8299", "#3E9FB3", "#7ABECC", "#B8DEE6", "#3D0F99", "#653EB3",
    "#967ACC", "#C7B8E6", "#333333", "#666666", "#999999", "#CCCCCC"
  ),
  parade = c(
    '#ff6969', '#9b37ff', '#cd3737', '#69cdff', '#ffff69', '#69cdcd',
    '#9b379b', '#3737cd', '#ffff9b', '#cdff69', '#ff9b37', '#37ffff',
    '#9b69ff', '#37cd69', '#ff3769', '#ff3737', '#37ff9b', '#cdcd37',
    '#3769cd', '#37cdff', '#9b3737', '#ff699b', '#9b9bff', '#cd9b37',
    '#69ff37', '#cd3769', '#cd69cd', '#cd6937', '#3737ff', '#cdcd69',
    '#ff9b69', '#cd37cd', '#9bff37', '#cd379b', '#cd6969', '#69ff9b',
    '#ff379b', '#9bff9b', '#6937ff', '#69cd37', '#cdff37', '#9bff69',
    '#9b37cd', '#ff37ff', '#ff37cd', '#ffff37', '#37cd9b', '#379bff',
    '#ffcd37', '#379b37', '#ff9bff', '#379b9b', '#69ffcd', '#379bcd',
    '#ff69ff', '#ff9b9b', '#37ff69', '#ff6937', '#6969ff', '#699bff',
    '#ffcd69', '#69ffff', '#37ff37', '#6937cd', '#37cd37', '#3769ff',
    '#cd69ff', '#6969cd', '#9bcd37', '#69ff69', '#37cdcd', '#cd37ff',
    '#37379b', '#37ffcd', '#69cd69', '#ff69cd', '#9bffff', '#9b9b37'
  )
)
seurat_continuous_palettes <- list(
  seurat = hue_pal()(16),
  seurat.16 = hue_pal()(16),
  seurat.32 = hue_pal()(32),
```

```
  seurat.64 = hue_pal()(64)
)
stripe_palettes <- list(
  stripe = rep(c("white", "grey60"), 8),
  stripe.16 = rep(c("white", "grey60"), 8),
  stripe.32 = rep(c("white", "grey60"), 16),
  stripe.64 = rep(c("white", "grey60"), 32)
)
tableau_palettes <- list()
orig_tableau_palettes <- ggthemes::ggthemes_data[["tableau"]][["color-palettes"]]
for (g in names(orig_tableau_palettes)) {
  for (pal in names(orig_tableau_palettes[[g]])) {
    palcolors <- as.list(orig_tableau_palettes[[g]][[pal]])
    if (!is.null(palcolors$name)) {
      tableau_palettes[[pal]] <- stats::setNames(palcolors$value, palcolors$name)
    } else {
      tableau_palettes[[pal]] <- palcolors$value
    }
  }
}

palette_list <- list()
all_colors <- c(
  rownames(brewer.pal.info), names(ggsci_db), rownames(redmonder.pal.info),
  rownames(metacartocolors), names(nord_palettes), names(viridis_palettes),
 ocean_names, names(dichromat_palettes), jcolors_names, names(seurat_discrete_palettes),
  names(seurat_continuous_palettes), custom_names, names(stripe_palettes),
  names(tableau_palettes)
)
for (pal in all_colors) {
  if (!pal %in% all_colors) {
    stop(paste0("Invalid pal Must be one of ", paste0(all_colors, collapse = ",")))
  }
  if (pal %in% rownames(brewer.pal.info)) {
    pal_n <- brewer.pal.info[pal, "maxcolors"]
    pal_category <- brewer.pal.info[pal, "category"]
    if (pal_category == "div") {
      palcolor <- rev(brewer.pal(name = pal, n = pal_n))
    } else {
      if (pal == "Paired") {
        palcolor <- brewer.pal(12, "Paired")[c(1:4, 7, 8, 5, 6, 9, 10, 11, 12)]
      } else {
        palcolor <- brewer.pal(name = pal, n = pal_n)
      }
    }
    if (pal_category == "qual") {
      attr(palcolor, "type") <- "discrete"
    } else {
      attr(palcolor, "type") <- "continuous"
    }
  } else if (pal %in% names(ggsci_db)) {
    if (pal %in% c("d3", "uchicago", "material")) {
      for (subpal in names(ggsci_db[[pal]])) {
```

```
      palcolor <- ggsci_db[[pal]][[subpal]]
      if (pal == "material") {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
      palette_list[[paste0(pal, "-", subpal)]] <- palcolor
    }
    next
  } else {
    palcolor <- ggsci_db[[pal]][[1]]
    if (pal == "gsea") {
      attr(palcolor, "type") <- "continuous"
    } else {
      attr(palcolor, "type") <- "discrete"
    }
  }
} else if (pal %in% rownames(redmonder.pal.info)) {
  pal_n <- redmonder.pal.info[pal, "maxcolors"]
  pal_category <- redmonder.pal.info[pal, "category"]
  if (pal_category == "div") {
    palcolor <- rev(redmonder.pal(name = pal, n = pal_n))
  } else {
    palcolor <- redmonder.pal(name = pal, n = pal_n)
  }
  if (pal_category == "qual") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% rownames(metacartocolors)) {
  pal_n <- metacartocolors[pal, "Max_n"]
  palcolor <- carto_pal(name = pal, n = pal_n)
  if (pal_category == "qualitative") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% names(nord_palettes)) {
  palcolor <- nord_palettes[[pal]]
  attr(palcolor, "type") <- "discrete"
} else if (pal %in% names(viridis_palettes)) {
  palcolor <- viridis_palettes[[pal]]
  attr(palcolor, "type") <- "continuous"
} else if (pal %in% names(ocean_palettes)) {
  palcolor <- ocean_palettes[[pal]]
  attr(palcolor, "type") <- "continuous"
} else if (pal %in% names(dichromat_palettes)) {
  palcolor <- dichromat_palettes[[pal]]
  if (pal %in% c("Categorical.12", "SteppedSequential.5")) {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
```

```
    }
  } else if (pal %in% jcolors_names) {
    palcolor <- jcolors(palette = gsub("jcolors-", "", pal))
    if (pal %in% paste0("jcolors-", c("pal10", "pal11", "pal12", "rainbow"))) {
      attr(palcolor, "type") <- "continuous"
    } else {
      attr(palcolor, "type") <- "discrete"
    }
  } else if (pal %in% custom_names) {
    palcolor <- custom_palettes[[pal]]
    if (pal %in% c("jet")) {
      attr(palcolor, "type") <- "continuous"
    } else {
      attr(palcolor, "type") <- "discrete"
    }
  } else if (pal %in% names(seurat_discrete_palettes)) {
    palcolor <- seurat_discrete_palettes[[pal]]
    attr(palcolor, "type") <- "discrete"
  } else if (pal %in% names(seurat_continuous_palettes)) {
    palcolor <- seurat_continuous_palettes[[pal]]
    attr(palcolor, "type") <- "continuous"
  } else if (pal %in% names(stripe_palettes)) {
    palcolor <- stripe_palettes[[pal]]
    attr(palcolor, "type") <- "discrete"
  } else if (pal %in% names(tableau_palettes)) {
    palcolor <- tableau_palettes[[pal]]
    attr(palcolor, "type") <- "discrete"
  }
  palette_list[[pal]] <- palcolor
 }
}

## End(Not run)
```

---

palette_this *Color palettes collected in plotthis.*

Description

Color palettes collected in plotthis.

Usage

```
palette_this(
  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	A vector of character/factor or numeric values. If missing, numeric values 1:n will be used as x.
n	The number of colors to return for numeric values.
palette	Palette name. All available palette names can be queried with `show_palettes()`.
palcolor	Custom colors used to create a color palette.
type	Type of `x`. Can be one of "auto", "discrete" or "continuous". The default is "auto", which automatically detects if `x` is a numeric value.
keep_names	Whether to keep the names of the color vector.
alpha	The alpha value of the colors. Default is 1.
matched	If `TRUE`, will return a color vector of the same length as `x`.
reverse	Whether to invert the colors.
NA_keep	Whether to keep the color assignment to NA in `x`.
NA_color	Color assigned to NA if NA_keep is `TRUE`.
transparent	Whether to make the colors transparent when alpha < 1. When `TRUE`, `ggplot2::alpha()` is used to make the colors transparent. Otherwise, `adjcolors` is used to adjust the colors based on the alpha. The color will be not be actually transparent. For example, `ggplot2::alpha("red", 0.5) == "#FF000080"`; while `adjcolors("red", 0.5) == "#FF8080"`.

Value

A vector of colors.

`PieChart` *Pie chart*

Description

Draws a pie chart illustrating the numerical proportion of each group relative to the whole. Each slice corresponds to a level of the x-axis variable and its angle is proportional to the y-axis value (or the observation count when `y` is omitted).

The function supports **count aggregation** (omit `y` to plot observation counts per x-category), **slice labels** via `ggrepel::geom_label_repel()`, clockwise or counter-clockwise slice ordering, faceting, and splitting into separate sub-plots via `split_by`.

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_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  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.
x	A character string specifying the column name for the x-axis (categories). Must be character or factor. Each unique value becomes a pie slice.
y	A character string specifying the numeric column for the y-axis. When NULL (default), the count of observations in each (x, facet_by) combination is used

	and stored as `.y`.
`label`	A character string specifying the column to use for slice labels. `NULL` (default) hides labels. When `TRUE`, the `y` values are used as labels. When `y = NULL`, use `".y"` to label with the computed counts.
`split_by`	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with `split_by_sep`.
`split_by_sep`	A character string to separate concatenated `split_by` columns. Default `"_"`.
`clockwise`	A logical value. When `TRUE` (default), the pie slices are ordered clockwise starting from the top. When `FALSE`, slices are ordered counter-clockwise.
`facet_by`	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`
`facet_scales`	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`
`facet_ncol`	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
`facet_nrow`	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
`facet_byrow`	A logical value indicating whether to fill the plots by row. Default is TRUE.
`theme`	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
`theme_args`	A list of arguments to pass to the theme function.
`palette`	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.
`palcolor`	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
`palreverse`	A logical value indicating whether to reverse the palette. Default is FALSE.
`alpha`	A numeric value specifying the transparency of the plot.
`aspect.ratio`	A numeric value specifying the aspect ratio of the plot.
`keep_na`	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.
`keep_empty`	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

combine
: Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow
: Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow
: Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

seed
: A numeric seed for reproducibility. Passed to validate_common_args().

axes
: A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).

axis_titles
: A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.

guides
: A character string specifying how guides (legends) should be collected across panels. Default "collect" (passed to combine_plots()).

design
: A custom layout design for the combined plot (passed to combine_plots()).

...
: Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. validate_common_args() validates the seed and facet_by settings.
2. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (x, split_by, facet_by).
3. process_theme() resolves the theme function.

4. The x column is forced to factor; y is validated.
5. The split_by column is validated and its NA / empty levels are processed via process_keep_na_empty(). It is then removed from the per-column keep_na / keep_empty lists.
6. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
7. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
8. PieChartAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
9. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
data <- data.frame(
   x = factor(c("A", "B", "C", NA, "E", "F", "G", "H"), levels = LETTERS[1:8]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = factor(c("G1", "G1", NA, NA, "G3", "G3", "G4", "G4"),
       levels = c("G1", "G2", "G3", "G4")),
   facet = factor(c("F1", NA, "F3", "F4", "F1", NA, "F3", "F4"),
       levels = c("F1", "F2", "F3", "F4"))
)

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

# Keep NA and empty levels
PieChart(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE)

# Counter-clockwise ordering
PieChart(data, x = "x", y = "y", clockwise = FALSE)
PieChart(data, x = "x", y = "y", clockwise = FALSE,
         keep_na = TRUE, keep_empty = TRUE)

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

# Faceting
PieChart(data, x = "x", y = "y", facet_by = "facet")
PieChart(data, x = "x", y = "y", facet_by = c("facet", "group"),
    keep_empty = "level")
PieChart(data, x = "x", y = "y", facet_by = c("facet", "group"),
    keep_empty = TRUE)

# Split into sub-plots
PieChart(data, x = "x", y = "y", split_by = "group")

# Per-split palettes
PieChart(data, x = "x", y = "y", split_by = "group",
         palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"))
```

```
# Y from count
PieChart(data, x = "group")

# Y from count with label
PieChart(data, x = "group", label = ".y")
```

`QQPlot` *QQ/PP plot*

Description

Produces a quantile-quantile (QQ) plot or probability-probability (PP) plot to compare the empirical distribution of a numeric variable against a theoretical distribution (default: standard normal). The function delegates to the **qqplotr** package for the underlying statistics and rendering.

Key features:

- **QQ and PP modes** – switch between quantile-quantile and probability-probability displays via `type`.
- **Confidence bands** – overlay one or more confidence bands (pointwise, KS, Tukey simultaneous, or bootstrap) with custom fill colours and alpha.
- **Reference line** – a diagonal reference line (QQ) or diagonal probability line (PP) for comparison.
- **Distribution fitting** – compare against any distribution supported by **qqplotr** (normal, exponential, uniform, etc.) by passing `distribution` and `dparams` inside the `band`, `line`, and `point` lists.
- **Detrending** – enable `detrend = TRUE` inside the argument lists to remove the reference line and visualise only deviations (flat PP plot centred at zero).
- **Splitting** – use `split_by` to produce separate QQ/PP plots for different groups, combined into a single layout.

Usage

```
QQPlot(
  data,
  val,
  val_trans = NULL,
  type = c("qq", "pp"),
  split_by = NULL,
  split_by_sep = "_",
  band = NULL,
  line = list(),
  point = list(),
  fill_name = "Bands",
  band_alpha = 0.5,
```

```
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  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,
  xlim = NULL,
  ylim = NULL,
  xlab = ifelse(type == "qq", "Theoretical Quantiles", "Probability Points"),
  ylab = ifelse(type == "qq", "Sample Quantiles", "Cumulative 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.
val	A character string naming the numeric column whose distribution is compared against the theoretical distribution.
val_trans	A transformation function applied to the val column before plotting. For example, log or sqrt. Default: NULL (no transformation).
type	A character string specifying the plot type. Either "qq" (quantile-quantile, the default) or "pp" (probability-probability). Partial matching is supported.
split_by	The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.
split_by_sep	The separator for multiple split_by columns. See split_by

band — A list of arguments passed to stat_qq_band or stat_pp_band, depending on type. Set to TRUE or an empty list to use default arguments. Set to NULL (the default) to suppress bands entirely. To add multiple bands, provide a list of lists, each containing arguments for one band (e.g. different bandType or distribution). Each band can also include a custom mapping aesthetic to control its fill colour legend entry.

line — A list of arguments passed to stat_qq_line or stat_pp_line, depending on type. Default: list() (adds a reference line with default arguments). Set to NULL to omit the line entirely.

point — A list of arguments passed to stat_qq_point or stat_pp_point, depending on type. Default: list() (adds points with default arguments). Set to NULL to omit points (not recommended).

fill_name — A character string for the fill legend title used when bands are present. Default: "Bands".

band_alpha — A numeric value in [0, 1] setting the transparency of all bands. Individual bands can override this via alpha inside the band argument list. Default: 0.5.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

`title` A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

`subtitle` A character string specifying the subtitle of the plot.

`xlim` A numeric vector of length 2 specifying the x-axis limits. Default: `NULL` (use data range).

`ylim` A numeric vector of length 2 specifying the y-axis limits. Default: `NULL` (use data range).

`xlab` A character string specifying the x-axis label.

`ylab` A character string specifying the y-axis label.

`combine` Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow` A numeric value specifying the number of rows in the facet.

`ncol` A numeric value specifying the number of columns in the facet.

`byrow` A logical value indicating whether to fill the plots by row.

`seed` The random seed to use. Default is 8525.

`axes` A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object (single plot), a patchwork object (combined split plots), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. **Common arg validation** – validate_common_args() checks the seed and facet_by constraints.
2. **Theme processing** – process_theme() resolves the theme string or function.
3. **split_by column resolution** – check_columns() validates the split_by column(s) with force_factor = TRUE. Multiple columns are concatenated with split_by_sep.
4. **Data splitting** – the data frame is split by split_by levels (droplevels applied, level order preserved). If split_by is NULL, the data is wrapped in a single-element list with name "...".
5. **Per-split parameter resolution** – check_palette(), check_palcolor(), and check_legend() resolve per-split palette, palcolor, legend.position, and legend.direction.
6. **Dispatch per split** – QQPlotAtomic() is called for each split level. If title is a function, it receives the split level name and generates a dynamic title; otherwise the level name is used as the default title.
7. **Combination** – results are combined via combine_plots() (when combine = TRUE) or returned as a named list (when combine = FALSE).

Examples

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

# Basic QQ plot with default confidence band
QQPlot(data, val = "norm", band = TRUE)

# Multiple confidence bands with custom fill labels
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)

# Compare against exponential distribution
data(airquality, package = "datasets")
di <- "exp"
dp <- list(rate = 2)
QQPlot(airquality, val = "Ozone",
    band = list(distribution = di, dparams = dp),
    line = list(distribution = di, dparams = dp),
    point = list(distribution = di, dparams = dp)
)
```

```
# Detrended QQ plot: deviations from the reference line
de <- TRUE
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)
)

# PP plot (probability-probability)
QQPlot(data, val = "norm", type = "pp", band = TRUE)

# PP plot with shifted/scaled normal distribution
dp <- list(mean = 2, sd = 2)
QQPlot(data, val = "norm", type = "pp",
    band = list(dparams = dp),
    point = list(dparams = dp))

# PP plot with custom intercept/slope line
QQPlot(data, val = "norm", type = "pp", band = TRUE,
    line = list(ab = c(.2, .5)))

# Detrended PP plot with axis limits
di <- "exp"
dp <- list(rate = .022)
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)
)
```

`RadarPlot` *Radar plot / Spider plot*

Description

Draws a radar chart (concentric circular grid) or spider chart (polygonal grid) displaying multivariate data in a two-dimensional polar coordinate system. Each x-axis category is placed at an evenly spaced angular position around the chart, and numeric values are plotted along the radial axis.

The function supports **count aggregation** (omit `y` to plot observation counts), **proportion scaling** (via `scale_y`), per-group colour control, faceting, and splitting into separate sub-plots via `split_by`.

`SpiderPlot` is an alias that renders the same data with polygonal grid lines (spider chart style) by using `polygon = TRUE`.

`SpiderPlot` is a variant of `RadarPlot` that renders the chart with straight polygonal grid lines (spider chart) instead of concentric circles. Internally, it calls `RadarPlotAtomic` with `polygon = TRUE` but is otherwise identical to `RadarPlot` in behaviour and parameters.

Usage

```
RadarPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  groups = 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 = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  keep_na = FALSE,
  keep_empty = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
```

```
  design = NULL,
  ...
)

SpiderPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  groups = 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 = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  keep_na = FALSE,
  keep_empty = FALSE,
  title = NULL,
  subtitle = 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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string used to join multiple x columns. Default "_". Ignored when x is a single column.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_name	A character string used as the colour/fill legend title. When NULL, the group_by column name is used.
groups	A character vector of group values (in the group_by column) to include in the plot. When NULL, all groups are included. This can control which groups appear and their legend order. Implies keep_empty = FALSE for the group_by column: groups not present in the data are not shown in the legend.
scale_y	How should the radial axis be scaled? Default is "group". Options are "group", "global", "x", and "none". • "group" — scaled to the fraction within each group. • "global" — scaled to the fraction of the total. • "x" — scaled to the fraction within each x-axis category. • "none" — raw counts or values, no scaling.
y_min	A numeric value setting the minimum of the radial axis. Default 0.
y_max	A numeric value setting the maximum of the radial axis. When NULL, the maximum data value is used.
y_nbreaks	A numeric value for the number of breaks (concentric grid lines) on the radial axis. Default 4.
bg_color	A character string specifying the background fill colour. Default "grey80".
bg_alpha	A numeric value for the transparency of the background fill. Default 0.1.
fill	A logical value. When TRUE (default), the data polygons are filled with the group colour. When FALSE, only outlines are drawn.
linewidth	A numeric value for the width of the polygon outline lines. Default 1.
pt_size	A numeric value for the size of the data point markers. Default 4.

max_charwidth A numeric value for the maximum character width of x-axis labels before wrapping. Default 16.

split_by The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.

split_by_sep A character string to separate concatenated split_by columns. Default "_".

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

alpha A numeric value specifying the transparency of the plot.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

keep_na A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
seed	A numeric seed for reproducibility. Passed to validate_common_args().
combine	Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.
ncol, nrow	Integer number of columns / rows for the combined layout (passed to wrap_plots).
byrow	Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).
axes	A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides (legends) should be collected across panels (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).
...	Additional arguments.

Value

A ggplot object (when combine = TRUE and split_by is NULL), a patchwork object (when combine = TRUE and split_by is provided), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

A ggplot object (when combine = TRUE and split_by is NULL), a patchwork object (when combine = TRUE and split_by is provided), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all relevant columns (x, split_by, group_by, facet_by).
2. The split_by column is validated and its NA / empty levels are processed via process_keep_na_empty(). It is then removed from the per-column keep_na / keep_empty lists.
3. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().

5. RadarPlotAtomic() is called for each split with polygon = FALSE. If title is a function, it receives the split level name and can generate dynamic titles.
6. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
set.seed(8525)

# --- Radar chart with observation counts ---
data <- data.frame(
    x = factor(
        c(rep("A", 20), rep("B", 30), rep(NA, 30), rep("D", 40), rep("E", 50)),
        levels = LETTERS[1:5]
    ),
    group = factor(
        sample(c("G1", NA, "G3", "G4"), 170, replace = TRUE),
        levels = c("G1", "G2", "G3", "G4")
    )
)

# Basic radar chart
RadarPlot(data, x = "x")

# Keep NA and empty factor levels
RadarPlot(data, x = "x", keep_na = TRUE, keep_empty = TRUE)

# Custom background colour
RadarPlot(data, x = "x", bg_color = "lightpink")

# Raw counts (no proportion scaling)
RadarPlot(data, x = "x", scale_y = "none")

# Grouped by a variable
RadarPlot(data, x = "x", group_by = "group", keep_na = TRUE)

# Faceted by a variable
RadarPlot(data, x = "x", facet_by = "group")

# Spider chart variant (polygonal grid)
SpiderPlot(data, x = "x")
SpiderPlot(data, x = "x", group_by = "group")

# --- Radar chart with explicit 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)
)

# Grouped radar with raw values
RadarPlot(data, x = "x", y = "y", scale_y = "none", group_by = "group")
```

```
# Faceted radar
RadarPlot(data, x = "x", y = "y", facet_by = "group")

# Split into separate sub-plots
RadarPlot(data, x = "x", y = "y", split_by = "group")

# Per-split palettes
RadarPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(G1 = "Set1", G2 = "Paired"))
```

`RarefactionPlot` *Rarefaction / extrapolation plot*

Description

Draws rarefaction and extrapolation curves for biodiversity data using the `iNEXT` package. Accepts raw species-abundance / incidence-frequency lists (which are passed to `iNEXT()` for estimation) or pre-computed `iNEXT` objects.

The function supports three curve types (sample-size-based, sample completeness, and coverage-based), diversity orders (`q`), per-group colouring, faceting, and splitting into separate sub-plots via `split_by`. Observed data are marked with points, rarefaction lines are solid, and extrapolation segments are dashed. Confidence intervals are shown as semi-transparent ribbons.

Usage

```
RarefactionPlot(
  data,
  type = 1,
  se = NULL,
  group_by = "group",
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  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	A data frame.
type	An integer specifying the curve type: 1 for sample-size-based rarefaction/extrapolation, 2 for sample completeness, or 3 for coverage-based rarefaction/extrapolation. A vector of types can be passed and the data will be fortifed for all of them; faceting or splitting then separates the panels. Default: 1.
se	A logical value indicating whether to display confidence intervals as semi-transparent ribbons around the estimated curve. When NULL (the default), it resolves to TRUE if the fortifed data contains y.lwr and y.upr columns, and FALSE otherwise.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	A character string used to join multiple group_by column values when group_by has length > 1. Also used by the exported function for the group concatenation. Default: "_".
group_name	A character string used as the title for the colour (and shape) legend. When NULL (the default), the value of group_by is used.
split_by	A character vector specifying how to split the data into separate sub-plots. Must be one or both of "q" (diversity order) and "group" (assemblage/site). Multiple values are concatenated with split_by_sep. Cannot overlap with group_by or facet_by. Default: NULL.
split_by_sep	A character string used to join multiple split_by column values when split_by has length > 1. Default: "_".
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

pt_size — A numeric value specifying the size of the observed-data points. Default: 3.

line_width — A numeric value specifying the width of the rarefaction / extrapolation lines. Default: 1.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

seed — A numeric seed for reproducibility. Passed to validate_common_args(). Default: 8525.

combine — Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow — Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow — Logical; fill the combined layout by row. Default TRUE.

axes — A character string specifying how axes should be treated across the combined layout (passed to combine_plots()).

axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how guides (legends) should be collected across panels (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).
...	Additional arguments passed to iNEXT when data is not already an iNEXT object. Common options include q (diversity order, default c(0, 1, 2)), datatype ("abundance" or "incidence"), and nboot (number of bootstrap replicates).

Value

A ggplot object (single split), a patchwork object (multiple splits with combine = TRUE), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. validate_common_args() checks the seed and facet_by validity.
2. The type argument is validated (must be one or more of 1, 2, 3).
3. group_by, split_by, and facet_by are validated for allowed values ("q" and/or "group") and checked for mutual exclusivity — no parameter may overlap with another.
4. If data is not an iNEXT object, it is passed to iNEXT() with ... (which may contain q, datatype, nboot, etc.).
5. The iNEXT object is fortifed via fortify() for the requested types. Columns Assemblage and Order.q are renamed to group and q, respectively.
6. The se parameter is resolved: if NULL it becomes TRUE when the fortifed data contains y.lwr / y.upr columns.
7. A lty column is created (factor with levels "Rarefaction" and "Extrapolation") to distinguish the two line phases via solid / dashed linetypes.
8. group_by, split_by, and facet_by are processed via check_columns() with force_factor = TRUE and multi-column concatenation.
9. If group_by is NULL, a dummy ".group" column is created and the legend is hidden.
10. The data is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
11. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
12. RarefactionPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
13. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

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))
)

# Basic sample-size-based rarefaction (type = 1)
RarefactionPlot(spider)

# Multiple diversity orders with faceting
RarefactionPlot(spider, q = c(0, 1, 2), facet_by = "q")

# Multiple diversity orders split into sub-plots
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q")

# Per-split palettes
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q",
                palette = c("0" = "Paired", "1" = "Set1", "2" = "Dark2"))

# Coverage-based rarefaction (type = 3) with
# group_by = "q" and facet_by = "group"
RarefactionPlot(spider, q = c(0, 1, 2), group_by = "q",
 facet_by = "group", palette = "Set1", type = 3)
```

`RidgePlot` *Ridge Plot*

Description

Ridge (joy) plot for visualising the distribution of a numeric variable across multiple groups. Each group is rendered as a partially overlapping density curve along the y-axis, making it easy to compare distribution shapes, central tendency, and spread across categories.

The function supports both **long** and **wide** data formats:

- **Long form** (`in_form = "long"`, default) — a numeric column (`x`) plus a factor column (`group_by`) whose levels become the y-axis ridges.
- **Wide form** (`in_form = "wide"`) — multiple numeric columns listed in `group_by` are gathered internally into long form.

Optional vertical reference lines (`add_vline`) can mark group means, specific values, or per-group thresholds. Supports faceting, split-by splitting, and full palette customisation.

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,
  keep_na = FALSE,
  keep_empty = FALSE,
  add_vline = NULL,
  vline_type = "solid",
  vline_color = TRUE,
  vline_width = 0.5,
  vline_alpha = 1,
  flip = FALSE,
  alpha = 0.8,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  x_text_angle = 90,
  x_min = NULL,
  x_max = NULL,
  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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
in_form	A character string specifying whether data is in "long" (default) or "wide" format.
split_by	The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the legend title for the group_by fill aesthetic. Defaults to the (concatenated) group_by column name.
scale	A numeric value controlling the vertical overlap of ridges. Passed to ggridges::geom_density_ridges(= ...). Smaller values increase overlap. When NULL, ggridges auto-computes the scale.
keep_na	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot.

- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

add_vline — A specification for vertical reference lines:

- NULL or FALSE: no lines.
- TRUE: draw a line at the mean of each group.
- A numeric vector: draw the same lines for all groups.
- A named list of numeric vectors: per-group lines, where names should match group_by levels.

vline_type — A character string specifying the line type for the vertical reference lines. Passed as linetype to geom_vline(). Default: "solid".

vline_color — The colour of the vertical reference lines:

- A literal colour value or vector (recycled): applied directly.
- TRUE (default): each line is coloured with a darkened blend of the corresponding ridge fill colour, computed via blend_colors(mode = "multiply").

vline_width — A numeric value for the thickness of the vertical reference lines. Passed as linewidth to geom_vline(). Default: 0.5.

vline_alpha — A numeric value in [0, 1] for the transparency of the vertical reference lines. Default: 1.

flip — A logical value. If TRUE, the axes are swapped via coord_flip(). X-axis text angle and grid-line placement are adjusted accordingly.

alpha — A numeric value specifying the transparency of the plot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

x_text_angle — A numeric value specifying the angle of the x-axis text.

x_min, x_max — Numeric limits for the x-axis. When NULL (default), limits are determined from the data range. Passed to scale_x_continuous(limits = ...).

reverse
: A logical value. If TRUE, the y-axis group order is reversed. NA groups are renamed to the literal string "NA" and placed at the end.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

seed
: The random seed to use. Default is 8525.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.

- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A `ggplot` object (single plot), a `patchwork` / `wrap_plots` object (when `split_by` is provided and `combine = TRUE`), or a list of `ggplot` objects (when `split_by` is provided and `combine = FALSE`).

split_by Workflow

When `split_by` is specified, `RidgePlot()` executes the following pipeline:

1. **Argument validation** — `validate_common_args()` checks the seed and facet-by consistency.
2. **NA / empty normalisation** — `check_keep_na()` / `check_keep_empty()` convert `keep_na` / `keep_empty` to per-column lists.
3. **Theme resolution** — `process_theme()` resolves the theme string to a theme function.
4. **Split column resolution** — `check_columns()` validates `split_by` (force_factor, concat_multi).
5. **Pre-filtering** — `process_keep_na_empty()` removes NA / empty levels from the split column, then `data` is split by `split_by` levels (order preserved).
6. **Per-split parameter resolution** — `check_palette()`, `check_palcolor()`, `check_legend()` resolve palette, palcolor, legend.position, and legend.direction for each split.
7. **Per-split dispatch** — each split is passed to `RidgePlotAtomic()` with its resolved parameters. Title defaults to the split level name unless `title` is a function (in which case it is called with the default).
8. **Combination** — `combine_plots()` assembles the list of plots via `patchwork::wrap_plots()`, applying `nrow`, `ncol`, `byrow`, `axes`, `axis_titles`, `guides`, and `design`.

Examples

```
set.seed(8525)
data <- data.frame(
   x = c(rnorm(250, -1), rnorm(250, 1)),
   group = factor(rep(c("A", NA, LETTERS[3:5]), each = 100), levels = LETTERS[1:6])
)

# basic usage
RidgePlot(data, x = "x")  # single ridge (no group_by)
RidgePlot(data, x = "x", add_vline = 0, vline_color = "black")

# grouped ridges
RidgePlot(data, x = "x", group_by = "group")
RidgePlot(data, x = "x", group_by = "group",
```

```
   keep_na = TRUE, keep_empty = TRUE)
RidgePlot(data, x = "x", group_by = "group", reverse = TRUE)
RidgePlot(data, x = "x", group_by = "group",
   add_vline = TRUE, vline_color = TRUE, alpha = 0.7)

# faceting
RidgePlot(data, x = "x", facet_by = "group",
   keep_na = TRUE, keep_empty = 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")

# split_by with per-split palettes
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", split_by = "group",
   palette = list(a = "Reds", b = "Blues", c = "Greens", d = "Purples"))
```

`RingPlot` *Ring plot (multi-layer donut chart)*

Description

Draws a ring plot (multi-layer donut chart) where each level of `x` becomes a concentric ring divided into filled segments by `group_by`. The plot is built with `geom_col()` under `coord_polar("y")`, producing a publication-quality ring chart with automatic count aggregation, per-group colour assignment, faceting, and splitting into sub-plots.

When `x = NULL`, a single-ring plot is produced (functionally equivalent to a pie chart via `PieChart`).

Usage

```
RingPlot(
  data,
  x = NULL,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  label = NULL,
  split_by = NULL,
  split_by_sep = "_",
```

```
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the fill legend title. When NULL, the group_by column name is used.
label	A logical value controlling whether ring labels are shown. Labels display the x values (ring names) at the inner edge of each ring. Default NULL auto-selects: FALSE for single-ring plots, TRUE for multi-ring plots.

split_by — The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.

split_by_sep — A character string to separate concatenated split_by columns. Default "_".

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

combine
: Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow
: Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow
: Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

seed
: A numeric seed for reproducibility. Passed to validate_common_args().

axes
: A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).

axis_titles
: A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.

guides
: A character string specifying how guides (legends) should be collected across panels. Default "collect" (passed to combine_plots()).

design
: A custom layout design for the combined plot (passed to combine_plots()).

...
: Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (x, group_by, split_by, facet_by).
2. The split_by column is validated and its NA / empty levels are processed via process_keep_na_empty(). It is then removed from the per-column keep_na / keep_empty lists.
3. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
5. RingPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
6. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

See Also

PieChart

Examples

```
# Basic single-ring plot (pie-chart-like)
RingPlot(datasets::iris, group_by = "Species")

# Multi-ring plot with faceting
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", facet_by = "vs")

# Split into sub-plots with per-split palettes
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", split_by = "vs",
        palette = c("0" = "Set1", "1" = "Paired"))

# Custom data with NA and empty levels
data <- data.frame(
  x = factor(c("A", "B", NA, "D", "A", "B", NA, "D"), levels = c("A", "B", "C", "D")),
  y = c(1, 2, 5, 3, 4, 5, 2, 6),
  group = factor(c("a", "a", "a", NA, NA, "c", "c", "c"), levels = c("a", "b", "c"))
)

# Default: NA and empty levels dropped
RingPlot(data, x = "x", y = "y", group_by = "group")

# Keep NA and empty levels
RingPlot(data, x = "x", y = "y", group_by = "group",
        keep_na = TRUE, keep_empty = TRUE)

# Per-column keep_na / keep_empty via named lists
RingPlot(data, x = "x", y = "y", group_by = "group",
        keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))
```

ROCCurve *ROC curve*

Description

Draws one or more Receiver Operating Characteristic (ROC) curves for evaluating binary classifier performance. The function wraps ROCCurveAtomic with split_by handling, providing the ability to generate separate ROC curves per split level and combine them via wrap_plots.

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,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  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.
truth_by	A character string naming the column that contains the true class labels (binary outcome, 0/1 or TRUE/FALSE).
score_by	A character vector of column names containing the predicted scores (classifier output values). When multiple columns are provided, each column becomes a separate ROC curve grouped by a .group identifier. When multiple columns are used, group_by must be NULL.
pos_label	A character string specifying the positive class label in truth_by. When NULL (default), the labels are handled by the plotROC package: if truth_by is a factor, the last level is used; otherwise it is coerced to a factor with a warning.
split_by	The column(s) to split the data by and produce separate ROC curve plots for each level. The split_by column is removed from the per-split data to avoid interfering with ROC analysis. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string used to separate concatenated split_by columns. Default: "_".
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to use as the legend title for the ROC curve groups. When NULL (default), the group_by column name is used.
x_axis_reverse	A logical value. If TRUE, the x-axis is reversed (from 1 to 0), displaying specificity instead of 1 - specificity. The x-axis label automatically changes to "Specificity". Default: FALSE.
percent	A logical value. If TRUE, the x and y axes are displayed as percentages (0 to 100). Default: FALSE.
ci	A list of arguments passed to plotROC::geom_rocci() to add confidence intervals to the ROC curve. When NULL (default), no confidence intervals are drawn. Example: ci = list(sig.level = 0.05).
n_cuts	An integer specifying the number of evenly-spaced quantile-based cutoff points to annotate on the ROC curve. Quantiles are computed from the score_by distribution. Default: 0 (no quantile cutoffs). Ignored when cutoffs_at is non-NULL.
cutoffs_at	A vector of user-supplied cutoff values to annotate as points on the ROC curve. When non-NULL, overrides n_cuts. Accepts raw numeric score thresholds and/or named method strings from the optimal.cutpoints package for automatic optimal cutoff identification. Both cutoffs_at and cutoffs.labels are passed to plotROC::geom_roc(). Supported method values are:

- "CB" (cost-benefit method);
- "MCT" (minimises Misclassification Cost Term);
- "MinValueSp" (a minimum value set for Specificity);
- "MinValueSe" (a minimum value set for Sensitivity);
- "ValueSe" (a value set for Sensitivity);
- "MinValueSpSe" (a minimum value set for Specificity and Sensitivity);
- "MaxSp" (maximises Specificity);
- "MaxSe" (maximises Sensitivity);
- "MaxSpSe" (maximises Sensitivity and Specificity simultaneously);
- "MaxProdSpSe" (maximises the product of Sensitivity and Specificity);
- "ROC01" (minimises distance between ROC plot and point (0,1));
- "SpEqualSe" (Sensitivity = Specificity);
- "Youden" (Youden Index);
- "MaxEfficiency" (maximises Efficiency/Accuracy);
- "Minimax" (minimises the most frequent error);
- "MaxDOR" (maximises Diagnostic Odds Ratio);
- "MaxKappa" (maximises Kappa Index);
- "MinValueNPV" (a minimum value set for Negative Predictive Value);
- "MinValuePPV" (a minimum value set for Positive Predictive Value);
- "ValueNPV" (a value set for Negative Predictive Value);
- "ValuePPV" (a value set for Positive Predictive Value);
- "MinValueNPVPPV" (a minimum value set for Predictive Values);
- "PROC01" (minimises distance between PROC plot and point (0,1));
- "NPVEqualPPV" (Negative Predictive Value = Positive Predictive Value);
- "MaxNPVPPV" (maximises Positive and Negative Predictive Values simultaneously);
- "MaxSumNPVPPV" (maximises the sum of the Predictive Values);
- "MaxProdNPVPPV" (maximises the product of Predictive Values);
- "ValueDLR.Negative" (a value set for Negative Diagnostic Likelihood Ratio);
- "ValueDLR.Positive" (a value set for Positive Diagnostic Likelihood Ratio);
- "MinPvalue" (minimises p-value of the Chi-squared test);
- "ObservedPrev" (closest value to observed prevalence);
- "MeanPrev" (closest value to the mean of the test values);
- "PrevalenceMatching" (predicted prevalence equals observed prevalence).

`cutoffs_labels` A character vector of user-supplied labels for the cutoff points. Must be the same length as `cutoffs_at`. When NULL, labels are generated automatically (score value or method name).

`cutoffs_accuracy`
A numeric value controlling the rounding precision of automatically generated cutoff labels. Default: `0.01`.

cutoffs_pt_size
: A numeric value specifying the size of the cutoff point markers. Default: 5.

cutoffs_pt_shape
: A numeric value specifying the shape of the cutoff point markers. Default: 4 (cross).

cutoffs_pt_stroke
: A numeric value specifying the stroke width of the cutoff point markers. Default: 1.

cutoffs_labal_fg
: A character string specifying the text colour of the cutoff labels. Default: "black".

cutoffs_label_size
: A numeric value specifying the font size of the cutoff labels. Default: 4.

cutoffs_label_bg
: A character string specifying the background colour of the cutoff labels. Default: "white".

cutoffs_label_bg_r
: A numeric value specifying the background radius of the cutoff labels (passed to ggrepel::geom_text_repel()). Default: 0.1.

show_auc
: A character string specifying the display mode for AUC values:
  - "auto" (default): Automatically determine the position. When there is a single group or facet_by is provided, AUC is placed on the plot; otherwise AUC is placed in the legend.
  - "none": Do not display AUC values.
  - "legend": Display AUC values in the legend labels.
  - "plot": Display AUC values as text on the plot.

auc_accuracy
: A numeric value controlling the rounding precision of AUC values in labels. Default: 0.01.

auc_size
: A numeric value specifying the font size of AUC labels when displayed on the plot. Default: 4.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

alpha
: A numeric value specifying the transparency of the plot.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
combine	A logical value. When TRUE (default), the list of per-split plots is combined into a single patchwork object with attr(p, "auc") and attr(p, "cutoffs") containing the aggregated results. When FALSE, returns a named list of individual ggplot objects.
nrow, ncol	Integer values specifying the number of rows and columns in the combined plot layout. Passed to wrap_plots.
byrow	A logical value. If TRUE (default), the combined layout is filled row-wise. Passed to wrap_plots.
seed	A numeric seed for reproducibility. Default: 8525. Passed to validate_common_args().
axes	A character string specifying how axes are treated across the combined layout. Passed to combine_plots(). Options: "keep", "collect", "collect_x", "collect_y".
axis_titles	A character string specifying how axis titles are treated across the combined layout. Defaults to axes. Passed to combine_plots().
guides	A character string specifying how legends are collected across panels in the combined layout. Passed to combine_plots().
design	A custom layout specification for the combined plot. Passed to combine_plots(). When specified, nrow, ncol, and byrow are ignored.
...	Additional arguments.

Details

Key features:

- **Multiple classifiers** — compare several prediction scores side-by-side by providing multiple `score_by` columns.
- **AUC display** — area under the curve values shown on the plot or in the legend, with configurable precision.
- **Optimal cutoffs** — identify and annotate optimal cutoff points using any of the 30+ methods from the `OptimalCutpoints` package, or supply custom numeric thresholds.
- **Confidence intervals** — add ROC confidence bands via `plotROC::geom_rocci()`.
- **Axis orientation** — reverse x-axis to show specificity or display axes as percentages.
- **Splitting and faceting** — split data into sub-plots via `split_by` or facet within a single plot via `facet_by`.

Value

A `patchwork` object (when `combine = TRUE`) with `attr(p, "auc")` and `attr(p, "cutoffs")` data frames containing aggregated AUC values and cutoff information across all splits. When `combine = FALSE`, returns a named list of `ggplot` objects, each with their own `attr(p[[i]], "auc")` and `attr(p[[i]], "cutoffs")`.

split_by Workflow

When `split_by` is provided, the following pipeline executes:

1. **Validation** — `validate_common_args()` checks the random seed and `facet_by` configuration.
2. **Column resolution** — `check_columns()` resolves `split_by` (force_factor, allow_multi, concat_multi).
3. **Data splitting** — Unused factor levels in `split_by` are dropped via `droplevels()`, and the data is split by `split_by` levels (preserving factor level order). If `split_by` is NULL, the data is wrapped in a single-element list named `"..."`.
4. **Per-split resolution** — `check_palette()`, `check_palcolor()`, and `check_legend()` resolve per-split palette, colour, legend.position, and legend.direction overrides.
5. **Per-split dispatch** — For each split:
   - Title resolution: if `title` is a function, it receives the split level name; otherwise `title %||% split_level` is used.
   - The `split_by` column is removed from the per-split data frame to avoid conflicts with the ROC analysis.
   - `ROCCurveAtomic()` is called with the per-split palette, palcolor, legend.position, and legend.direction.
6. **Combination** — `combine_plots()` assembles the list of plots via `patchwork::wrap_plots`, honouring `nrow`/`ncol`/`byrow`/`design`.
7. **AUC / cutoff collection** — When `combine = TRUE`, the per-split `auc` and `cutoffs` attributes are collected into combined data frames with a `split_by` column identifying the source split, and stored as `attr(p, "auc")` and `attr(p, "cutoffs")`.

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)

# --- Basic ROC curve ---
ROCCurve(data, truth_by = "D", score_by = "M1")

# --- Will warn about the positive label ---
ROCCurve(data, truth_by = "D.str", score_by = "M1")

# --- Decreasing direction ---
ROCCurve(data, truth_by = "D", score_by = "M1", increasing = FALSE)

# --- Multiple ROC curves (multiple classifiers) ---
ROCCurve(data, truth_by = "D", score_by = c("M1", "M2"), group_name = "Method")

# --- Grouping by a column ---
ROCCurve(data, truth_by = "D", score_by = "M1", group_by = "gender", show_auc = "plot")

# --- Reverse x-axis and display as percentages ---
ROCCurve(data, truth_by = "D", score_by = "M1", x_axis_reverse = TRUE, percent = TRUE)

# --- Custom n_cuts and single colour ---
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",
   cutoffs_at = c(0.2, "MaxSpSe"))
p
# Retrieve the AUC values
attr(p, "auc")
# Retrieve the cutoffs
attr(p, "cutoffs")
```

SankeyPlot *Sankey / Alluvial Plot*

Description

Draws Sankey (alluvial) diagrams to visualise flow, movement, or change from one categorical state to another across discrete positions (time points, stages, or groups). The plot consists of **nodes** (vertical blocks, or strata) representing categories at each position, and **links** (alluvia / flows) representing the observation units that move between categories across positions.

The function accepts data in several formats, controlled by `in_form`:

`"lodes"` / `"long"` Each row is an observation at one x-position, with columns for `x`, `stratum`, `alluvium`, and optionally `y`.

`"alluvia"` / `"wide"` Each row is an observation unit tracked across all positions; `x` columns represent the categories at each position. Converted internally via `to_lodes_form()`.

`"counts"` Numeric columns under each `x` represent frequencies. When the first element of `x` is `"."`, the `links_fill_by` values are injected as an additional first column of nodes, visualising the source distribution of flows.

`"auto"` **(default)** Automatically detects the format: numeric multi-column `x` → `"counts"`; multi-column `x` passing `is_alluvia_form` → `"alluvia"`; otherwise → `"lodes"`.

Supports **split_by** to produce separate sub-plots for different subsets of the data, **facet_by** for within-plot faceting, and independent styling of nodes and links (colours, alpha, borders, labels, and legend behaviour).

`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,
  palreverse = FALSE,
  links_alpha = 0.6,
  legend.box = "vertical",
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  flip = FALSE,
  theme = "theme_this",
  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,
palreverse = FALSE,
links_alpha = 0.6,
legend.box = "vertical",
x_text_angle = 0,
aspect.ratio = 1,
legend.position = "right",
legend.direction = "vertical",
flip = FALSE,
theme = "theme_this",
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_form	A character string specifying the input data format. One of "auto" (default), "long", "lodes", "wide", "alluvia", or "counts". "long" is an alias for "lodes"; "wide" is an alias for "alluvia". See the data parameter of SankeyPlot for format descriptions.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to join multiple x columns when in_form is "lodes" or auto-determined as lodes. Default "_".
y	A character string specifying the column name of the data frame to plot for the y-axis.
stratum	A character string specifying the column that defines the node categories at each x-axis position. Each unique value becomes a stratum (node block) at each x position. When NULL, defaults to links_fill_by. Multiple columns are concatenated with stratum_sep. Ignored in "alluvia" format.
stratum_sep	A character string to join multiple stratum columns. Default "_".
alluvium	A character string specifying the column that identifies individual flows (alluvia) across x-axis positions. Each unique value represents a single observational unit tracked across positions. When NULL in "counts" format, an auto-generated identifier is created. Multiple columns are concatenated with alluvium_sep. Ignored in "alluvia" format.
alluvium_sep	A character string to join multiple alluvium columns. Default "_".
split_by	The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.
split_by_sep	The separator for multiple split_by columns. See split_by
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot.

- `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

`flow` A logical value. When `FALSE` (default), `geom_alluvium()` is used for the links. When `TRUE`, `geom_flow()` is used instead, which draws the flows with a directional gradient between x positions.

`expand` The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`nodes_legend` Controls how the node legend is displayed. One of:

`"auto"` **(default)** Automatically determined: if `nodes_label = TRUE`, or if `stratum` is identical to `links_fill_by` with matching colours, the legend is hidden. Otherwise, overlapping stratum values across x positions are checked: any overlap produces a merged legend; no overlap produces separate legends per x position.

`"merge"` A single merged legend for all nodes.

`"separate"` One legend per x-axis position, generated via separate `scale_fill_manual()` layers.

`"none"` No node legend is shown.

`nodes_color` A character string specifying the border colour of the node (stratum) rectangles. Use the special value `".fill"` to match the border colour to the node fill colour. Default `"grey30"`.

`links_fill_by` A character string specifying the column that determines the fill colour of the links (alluvia / flows). When `NULL` in `"lodes"` format, defaults to `alluvium`. In `"counts"` format with the `"."` prefix, this parameter is required. Multiple columns are concatenated with `links_fill_by_sep`.

`links_fill_by_sep` A character string to join multiple `links_fill_by` columns. Default `"_"`.

`links_name` A character string for the legend title of the link fill scale. When `NULL` (default), the `links_fill_by` column name is used.

`links_color` A character string specifying the border colour of the links (alluvia / flows). Use the special value `".fill"` to match the link border colour to the link fill colour. Default `"gray80"`.

`nodes_palette` A character string specifying the colour palette for the node (stratum) fill. Passed to `palette_this()`. Default `"Paired"`.

`nodes_palcolor` A character vector of custom colours for the node fill, used as `palcolor` in `palette_this()`. When `NULL` (default), the palette colours are used directly.

nodes_alpha
: A numeric value in $[0, 1]$ controlling the transparency of the node (stratum) fill. Default 1.

nodes_label
: A logical value. When TRUE, stratum labels are drawn inside each node using geom_label() with StatStratum. Default FALSE.

nodes_label_miny
: A numeric value specifying the minimum y (frequency) threshold for displaying node labels. Nodes with y-values below this threshold are not labelled. Default 0.

nodes_width
: A numeric value (typically 0–1) specifying the width of the node (stratum) rectangles as a fraction of the x-axis spacing. Default 0.25.

links_palette
: A character string specifying the colour palette for the link fill. Passed to palette_this(). Default "Paired".

links_palcolor
: A character vector of custom colours for the link fill, used as palcolor in palette_this(). When NULL (default), the palette colours are used directly.

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

links_alpha
: A numeric value in $[0, 1]$ controlling the transparency of the link fill. Default 0.6.

legend.box
: A character string specifying the arrangement of legend boxes, either "vertical" (default) or "horizontal".

x_text_angle
: A numeric value specifying the angle of the x-axis text.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

flip
: A logical value. When TRUE, coord_flip() is applied to swap the x and y axes. Default FALSE.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object (single panel, no split_by), a patchwork object (when combine = TRUE and split_by is used), or a named list of ggplot objects (when combine = FALSE). Each plot carries height and width attributes in inches.

split_by workflow

When split_by is provided:

1. The split_by column(s) are validated and coerced to factors via check_columns(). Multi-column split_by is concatenated with split_by_sep.
2. Empty factor levels are dropped from split_by.
3. The data is split by split_by level (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. SankeyPlotAtomic() is called for each split, with title resolved per level (supports function-valued titles).
5. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
# Examples from ggalluvial datasets
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

Draws a scatter plot with optional size encoding, colour encoding (continuous gradient or discrete palette), point highlighting, and axis transformations. This is the user-facing wrapper around

ScatterPlotAtomic that adds split_by support (generating separate sub-plots per group) and combines them via patchwork.

Key features:

- **Variable point size** – size_by accepts either a numeric constant or a column name.
- **Colour modes** – numeric color_by produces a continuous gradient; factor/character color_by produces a discrete palette.
- **Colour scale trimming** – lower_quantile / upper_quantile (or explicit lower_cutoff / upper_cutoff) trim/clamp continuous colour scale extremes.
- **Border modes** – border_color can be a constant colour, TRUE (track the fill gradient), or omitted.
- **Point highlighting** – highlight accepts indices, rownames, logical TRUE, or a string expression.
- **Axis transformation** – xtrans / ytrans support log, sqrt, and other scale transformations.
- **Split sub-plots** – split_by produces one scatter plot per group level, combined into a single patchwork layout.

Usage

```
ScatterPlot(
  data,
  x,
  y,
  size_by = 2,
  size_name = NULL,
  color_by = NULL,
  color_name = NULL,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  palreverse = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  shape = 21,
  alpha = ifelse(shape %in% 21:25, 0.65, 1),
  border_color = "black",
  border_size = 0.5,
  highlight = NULL,
  highlight_shape = 16,
  highlight_size = 3,
  highlight_color = "red",
  highlight_alpha = 1,
  theme = "theme_this",
  theme_args = list(),
 palette = ifelse(!is.null(color_by) && !is.numeric(data[[color_by]]), "Paired",
    "Spectral"),
```

```
  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.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
size_by	Either a numeric constant (uniform dot size) or a character string naming a numeric column whose values control dot size via `scale_size_area(max_size = 6)`. Default: 2.
size_name	A character string for the size legend title. When NULL (default), the size_by column name is used. Ignored when size_by is a numeric constant.
color_by	A character string naming a column whose values control dot colour. Can be numeric (continuous gradient via `scale_fill_gradientn()` / `scale_color_gradientn()`) or factor/character (discrete palette via `scale_fill_manual()` / `scale_color_manual()`). For shapes 21–25, the colour is applied to the fill aesthetic. When NULL (default), all dots are rendered in a single colour derived from the palette.
color_name	A character string for the colour legend title. When NULL (default), the color_by column name is used.
lower_quantile, upper_quantile	Lower and upper quantiles for the continuous color/fill scale. The actual cutoffs are determined by these quantiles when lower_cutoff and upper_cutoff are NULL. Defaults: lower_quantile = 0, upper_quantile = 0.99.

lower_cutoff, upper_cutoff
: Explicit lower and upper cutoffs for the continuous color/fill scale. When NULL (the default), the cutoffs are determined by lower_quantile and upper_quantile via quantile. Values outside the [lower_cutoff, upper_cutoff] range are clamped (winsorized) to the nearest cutoff value.

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

split_by
: The column(s) to split data by and generate separate scatter plots for each level. The split column is processed before splitting; multiple columns are concatenated with split_by_sep.

split_by_sep
: A character string used to concatenate multiple split_by column values. Default: "_".

shape
: A numeric value specifying the point shape. Default: 21 (filled circle with border). Shapes 21–25 support separate fill and border colour aesthetics; all other shapes use a single colour aesthetic.

alpha
: A numeric value specifying the transparency of the plot.

border_color
: Controls the point border colour. For shapes 21–25:

  - "black" (default) – constant black border.
  - A colour string (e.g. "red", "#FF0000") – constant colour border.
  - TRUE – border colour tracks the color_by gradient / palette via scale_color_gradientn() / scale_color_manual().

  For shapes without a fill aesthetic (not 21–25), this parameter has no effect.

border_size
: A numeric value specifying the point border size. Default: 0.5.

highlight
: Specifies which points to highlight with an overlaid geom_point() layer. Accepted values:

  - NULL (default) – no highlighting.
  - TRUE – all points are highlighted.
  - A numeric vector – row indices of points to highlight.
  - A single character string – an R expression (e.g. "x > 0") that is parsed with rlang::parse_expr() and evaluated via filter() to select rows.
  - A character vector – rownames of points to highlight. An error is thrown if the data has no rownames.

highlight_shape
: A numeric value specifying the point shape for highlighted points. Default: 16 (filled circle). Shapes 21–25 use the fill aesthetic; other shapes use color.

highlight_size
: A numeric value specifying the size of highlighted points. Default: 3.

highlight_color
: A character string specifying the colour of highlighted points. Default: "red".

highlight_alpha
: A numeric value in [0, 1] specifying the transparency of highlighted points. Default: 1.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

combine — A logical value. If TRUE (the default), the list of per-split plots is combined into a single patchwork object via combine_plots(). If FALSE, returns the raw list of ggplot objects.

nrow, ncol, byrow — Integers controlling the layout of combined plots via combine_plots(). byrow = TRUE (default) fills the layout row-wise.

seed — The random seed for reproducibility. Passed to validate_common_args(). Default: 8525.

axes, axis_titles — Strings controlling how axes and axis titles are handled across combined plots. Passed to combine_plots(). See ?patchwork::wrap_plots for options ("keep", "collect", "collect_x", "collect_y").

guides — A string controlling guide collection across combined plots. Passed to combine_plots().

design — A custom layout specification for combined plots. Passed to combine_plots(). When specified, nrow, ncol, and byrow are ignored.

... — Additional arguments.

Value

A ggplot object (single plot), a patchwork object (when combine = TRUE with split_by), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. **Seed validation** – validate_common_args() sets the random seed for reproducibility.
2. **Theme resolution** – process_theme() resolves the theme string or function.
3. **Split column resolution** – check_columns() validates split_by (force_factor, allow_multi, concat_multi).
4. **Data splitting** – unused factor levels are dropped and the data is split into a named list (preserving factor level order). When split_by = NULL, a single-element list named "..." is used.
5. **Per-split palette / colour** – check_palette() and check_palcolor() resolve per-split palette and colour overrides.
6. **Per-split legend** – check_legend() resolves legend.position and legend.direction per split.
7. **Per-split title** – when title is a function, it receives the default title (the split level name) and can return a custom string; otherwise title %||% split_level is used.
8. **Dispatch** – each split subset is passed to ScatterPlotAtomic().
9. **Combination** – combine_plots() assembles the list of plots via patchwork::wrap_plots, honouring nrow/ncol/byrow/design.

Examples

```
set.seed(8525)

data <- data.frame(
   x = rnorm(20),
   y = rnorm(20),
   w = abs(rnorm(20)),
   t = sample(c("A", "B"), 20, replace = TRUE)
)

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

# --- Highlight points ---
ScatterPlot(data, x = "x", y = "y", highlight = 'x > 0')

# --- Size encoding (column name) ---
ScatterPlot(data, x = "x", y = "y", size_by = "w")

# --- Colour encoding (numeric gradient) ---
ScatterPlot(data, x = "x", y = "y", color_by = "w")
```

```
# --- Colour encoding (categorical) with border ---
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = "red")

# --- Border colour tracks fill gradient ---
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = TRUE)

# --- Shape without fill (single colour aesthetic) ---
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 shape = 1, palette = "Set1")

# --- split_by with per-split palcolor ---
ScatterPlot(data, x = "x", y = "y", split_by = "t",
            palcolor = list(A = "blue", B = "red"))

# --- Colour scale limits (quantile-based) ---
ScatterPlot(data, x = "x", y = "y", color_by = "w",
            lower_quantile = 0.1, upper_quantile = 0.9)

# --- Colour scale limits (explicit cutoffs) ---
ScatterPlot(data, x = "x", y = "y", color_by = "w",
            lower_cutoff = 0, upper_cutoff = 1)
```

`show_palettes` *Show the color palettes*

Description

This function displays color palettes using ggplot2.

Usage

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

Arguments

`palettes`	A list of color palettes. If `NULL`, uses default palettes.
`type`	A character vector specifying the type of palettes to include. Default is "discrete".

index	A numeric vector specifying the indices of the palettes to include. Default is NULL.
palette_names	A character vector specifying the names of the SCP palettes to include. Default is NULL.
return_names	A logical value indicating whether to return the names of the selected palettes. Default is TRUE.
return_palettes	A logical value indicating whether to return the colors of selected palettes. Default is FALSE.

Value

A list of palette names or a list of palettes.

See Also

palette_list

Examples

```
show_palettes(palettes = list(c("red", "blue", "green"), c("yellow", "purple", "orange")))
all_palettes <- show_palettes(return_palettes = TRUE)
names(all_palettes)
all_palettes[["simspec"]]
show_palettes(index = 1:10)
show_palettes(type = "discrete", index = 1:10)
show_palettes(type = "continuous", index = 1:10)
show_palettes(
    palette_names = c("Paired", "nejm", "simspec", "Spectral", "jet"),
    return_palettes = TRUE
)
```

SpatImagePlot *Spatial visualization functions for terra objects and point data*

Description

These functions provide publication-quality spatial visualizations built on **ggplot2**, supporting raster images, categorical masks, vector shapes, and point data from the **terra** package or standard data frames.

Usage

```
SpatImagePlot(
  data,
  ext = NULL,
  raster = NULL,
```

```
  raster_dpi = NULL,
  flip_y = TRUE,
  palette = "turbo",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_box",
  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,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_box",
  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,
  palreverse = 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_box",
  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,
  palreverse = 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_box",
  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,
  palreverse = 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_box",
  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,
  palreverse = 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_box",
  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 SpatRaster, SpatVector, or data.frame depending on the function. See individual function descriptions.
ext	A numeric vector of length 4 c(xmin, xmax, ymin, ymax) or a terra::SpatExtent object. Default is NULL (use full extent).
raster	Whether to rasterize for efficient rendering of large datasets. Default is NULL (auto-detect: rasterize when ncell > 1e6 or nrow > 1e6). Uses scattermore::geom_scattermore() for points.

raster_dpi A numeric vector of length 1 or 2 specifying the raster output resolution in pixels. Default is c(512, 512).

flip_y Whether to negate the y-coordinates so the axis labels can be displayed with reversed sign. Default is TRUE.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

alpha A numeric value specifying the transparency of the plot.

fill_name A character string for the fill legend title.

return_layer Whether to return a list of ggplot layers instead of a complete plot. Default is FALSE. When TRUE, the returned list has a "scales" attribute for layer conflict detection, allowing multiple spatial layers in a single custom ggplot.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

legend.position A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

seed The random seed to use. Default is 8525.

add_border Whether to add polygon borders around mask regions in SpatMasksPlot. Default is TRUE.

border_color A character string for the border color. Default is "black". When TRUE in SpatShapesPlot or SpatPointsPlot, the border maps to the same variable as fill.

border_size A numeric value for the border line width. Default is 0.5.

border_alpha A numeric value for the border transparency. Default is 1.

x A character string specifying the x coordinate column for SpatPointsPlot and SpatShapesPlot (when data is a data frame). Auto-detected from common column names when NULL.

y A character string specifying the y coordinate column for SpatPointsPlot and SpatShapesPlot (when data is a data frame). Auto-detected from common column names when NULL.

group A character string specifying the grouping column for SpatShapesPlot when data is a data.frame. Each unique value in this column defines a separate polygon.

fill_by A character string or vector specifying the column(s) to map to fill color in SpatShapesPlot. When multiple columns are provided (all must be numeric), the data is reshaped and faceted. When a single string that does not match a column name, it is treated as a fixed fill color.

highlight A character vector of row names to highlight, a filter expression string (e.g., 'cat == "A"'), or TRUE (highlight all). Highlighted points are overlaid with larger markers.

highlight_alpha
A numeric value for highlight transparency. Default is 1.

highlight_size A numeric value for the highlight marker size. Default is 1.

highlight_color
A character string for the highlight marker color. Default is "black".

highlight_stroke
A numeric value for the highlight stroke width (added to border_size). Default is 0.8.

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

color_by A character string or vector specifying the column(s) to map to point color in SpatPointsPlot. Multiple numeric columns trigger faceting.

size_by A character string specifying the column to map to point size in SpatPointsPlot. Mutually exclusive with size.

size A numeric value for a fixed point size in SpatPointsPlot. Alias for size_by when given a scalar.

lower_quantile, upper_quantile
Lower and upper quantiles for the continuous color/fill scale. The actual cutoffs are determined by these quantiles when lower_cutoff and upper_cutoff are NULL. Defaults: lower_quantile = 0, upper_quantile = 0.99.

lower_cutoff, upper_cutoff
Explicit lower and upper cutoffs for the continuous color/fill scale. When NULL (the default), the cutoffs are determined by lower_quantile and upper_quantile via quantile. Values outside the [lower_cutoff, upper_cutoff] range are clamped (winsorized) to the nearest cutoff value.

color_name A character string for the color legend title.

size_name A character string for the size legend title.

shape A numeric value (21–25 for border shapes) or character string specifying the point shape in SpatPointsPlot. Default is 16 (filled circle).

hex Whether to use hexagonal binning in SpatPointsPlot. Requires a numeric color_by. Default is FALSE.

hex_linewidth A numeric value for the hex bin border width. Default is 0.5.

hex_count Whether to show hex bin count as point opacity. Default is FALSE.

hex_bins A numeric value for the number of hex bins. Default is 50.

hex_binwidth A numeric value for the hex bin width. Default is NULL.

label Whether to show group labels in SpatPointsPlot. Default is FALSE. Requires a categorical color_by.

label_size A numeric value for the label text size. Default is 4.

label_fg A character string for the label text color. Default is "white".

label_bg A character string for the label background color. Default is "black".

label_bg_r A numeric value for the label background corner radius ratio. Default is 0.1.

label_repel Whether to repel labels from each other and from data points. Default is FALSE.

label_repulsion
A numeric value for the repulsion force. Default is 20.

label_pt_size A numeric value for the label anchor point size. Default is 1.

label_pt_color A character string for the label anchor point color. Default is "black".

label_segment_color
A character string for the label connector line color. Default is "black".

label_insitu Whether to place raw group names as labels instead of numeric indices. Default is FALSE.

label_pos A character string or function specifying how label positions are computed per group. Options: "median" (default), "mean", "center", "first", "last", "random", "min", "max", or a custom function that takes a numeric vector and returns a single value.

graph Graph/network edge data for SpatPointsPlot. Can be a square adjacency matrix with row/column names matching data, a data.frame with edge coordinates (see graph_x etc.), a column name or index, or "@graph" (extracts from the data attribute named "graph"). Default is NULL.

graph_x A character string for the x start coordinate column in the graph data.

graph_y A character string for the y start coordinate column in the graph data.

graph_xend A character string for the x end coordinate column in the graph data.

graph_yend A character string for the y end coordinate column in the graph data.

graph_value A character string for the edge weight column in the graph data.

edge_size A numeric vector of length 2 specifying the line width range for graph edges. Default is c(0.05, 0.5).

edge_alpha A numeric value for graph edge transparency. Default is 0.1.

edge_color A character string for the graph edge color. Default is "grey40".

Details

`SpatImagePlot` Render a `SpatRaster` as a raster image. Supports single-layer continuous values (with gradient fill) and 3-channel RGB data (with automatic color identity scaling).

`SpatMasksPlot` Render a `SpatRaster` as a categorical mask overlay. Zero-valued cells are treated as transparent background; optional polygon borders can be added around mask regions.

`SpatShapesPlot` Render spatial shapes (polygons) from a `SpatVector` or a data frame of vertex coordinates. Supports single and multi-column fill mapping with automatic faceting.

`SpatPointsPlot` Render spatial points from a data frame with support for color/size/fill mapping, hex binning, rasterized rendering, network/graph overlays, labels, and point highlighting.

Value

A `ggplot` object with `height` and `width` attributes when `return_layer = FALSE` (the default). When `return_layer = TRUE`, a list of `ggplot` layers with a `"scales"` attribute is returned. When faceted via multiple `fill_by` or `color_by` columns, a faceted `ggplot` is returned.

Rendering Pipeline for SpatImagePlot

1. **Extent cropping** — if `ext` is provided, the `SpatRaster` is cropped via `terra::crop()`. An error is raised if no cells remain within the extent.
2. **Auto-rasterization** — if the raster exceeds 1e6 cells (or `raster = TRUE`), the raster is aggregated via `terra::aggregate()` to a target resolution of `raster_dpi`.
3. **Y-axis flipping** — when `flip_y = TRUE`, the raster is flipped vertically via `.flip_y()` and its y-extent is negated so that axis labels can be displayed with reversed sign.
4. **RGB detection** — if the raster has exactly 3 layers, they are treated as red/green/blue channels: each channel is rescaled to 0–255 and combined into a hex color via `rgb()`. A `scale_fill_identity()` is used with no legend guide.
5. **Single-layer rendering** — otherwise, the raster is converted to an x/y/value data frame and rendered via `geom_raster()` with `scale_fill_gradientn()`.
6. **Layer return or full assembly** — if `return_layer = TRUE`, the list of layers (with a `"scales"` attribute set to `"fill"`) is returned. Otherwise, `.wrap_spatial_layers()` assembles a complete ggplot with `coord_sf(expand = 0)`, theme, labels, legend, and dimension attributes.

Rendering Pipeline for SpatMasksPlot

1. **Extent cropping** — if `ext` is provided, the `SpatRaster` is cropped via `terra::crop()`.
2. **Y-axis flipping** — when `flip_y = TRUE`, the raster is flipped vertically and its y-extent is negated.
3. **Background masking** — cells with value 0 are set to `NA` so they render as transparent via `na.value = "transparent"`.
4. **Raster layer** — the mask is converted to an x/y/value data frame and rendered via `geom_raster()` with a gradient fill scale.
5. **Optional borders** — when `add_border = TRUE`, the mask values are polygonized via `terra::as.polygons()`, converted to sf, and overlaid as unfilled `geom_sf()` with the specified `border_color`, `border_size`, and `border_alpha`.

6. **Layer return or full assembly** — if `return_layer = TRUE`, the layers are returned; otherwise, `.wrap_spatial_layers()` creates the complete ggplot.

Rendering Pipeline for SpatPointsPlot

1. **Column resolution** — x and y coordinates are resolved from the data, auto-detecting common column names (`"x"`, `"X"`, `"sdimx"`, etc. for x; `"y"`, `"Y"`, `"sdimy"`, etc. for y) when not explicitly provided.
2. **Extent cropping** — if `ext` is provided, rows outside the extent are filtered.
3. **Y-axis flipping** — when `flip_y = TRUE`, the y coordinate column is negated via `.flip_y()`.
4. **Multi-column faceting** — when `color_by` has multiple columns (all numeric), the data is reshaped to long format with a `.facet_var` column and faceted via `facet_plot()`.
5. **Cutoff winsorization** — for numeric `color_by`, values outside `[lower_cutoff, upper_cutoff]` (derived from quantiles or explicit values) are clamped to the nearest cutoff.
6. **Graph / network edges** — if `graph` is provided, edges are resolved from an adjacency matrix, data.frame, or data attribute, and rendered as `geom_segment()` segments with line width proportional to edge weight.
7. **Main point layer** — one of three rendering modes:

   Regular (`hex = FALSE`, `raster = FALSE`) — `geom_point()` with shape, size, color, and fill aesthetic mappings. Shapes 21–25 support separate fill and border colors.

   Hex (`hex = TRUE`) — `geom_hex()` or `stat_summary_hex()` for binned aggregation of numeric `color_by` values.

   Raster (`raster = TRUE`) — `scattermore::geom_scattermore()` for efficient rendering of large datasets (>1e6 rows).
8. **Highlight** — if `highlight` is specified, highlighted points are overlaid as larger, colored markers using `geom_point()` or `scattermore::geom_scattermore()`.
9. **Labels** — if `label = TRUE` and `color_by` is a categorical column, group centroid positions are computed via `aggregate()` with the `label_pos` function, and labels are rendered via `ggrepel::geom_text_repel()` with optional background styling and repulsion.
10. **Layer return or full assembly** — if `return_layer = TRUE`, the layers are returned; otherwise, `.wrap_spatial_layers()` creates the complete ggplot and `facet_plot()` is applied when multi-column faceting is active.

Examples

```
set.seed(8525)
# --- SpatImagePlot ---
# Generate a sample SpatRaster
r <- terra::rast(
    nrows = 50, ncols = 40, vals = runif(2000),
    xmin = 0, xmax = 40, ymin = 0, ymax = 50,
    crs = ""
)
SpatImagePlot(r)
SpatImagePlot(r, raster = TRUE, raster_dpi = 20)
SpatImagePlot(r, alpha = 0.5, theme = "theme_blank",
    theme_args = list(add_coord = FALSE), fill_name = "value")
```

```
SpatImagePlot(r, ext = c(0, 10, 0, 10), flip_y = FALSE, palette = "viridis")

# --- SpatMasksPlot ---
m <- terra::rast(
   nrows = 50, ncols = 40,
   vals = sample(c(1:5, NA), 2000, replace = TRUE, prob = c(rep(0.04, 5), 0.8)),
   xmin = 0, xmax = 40, ymin = 0, ymax = 50,
   crs = ""
)
SpatMasksPlot(m, border_color = "red")
SpatMasksPlot(m, ext = c(0, 15, 0, 20), add_border = FALSE,
    palreverse = TRUE, fill_name = "value")

# --- SpatShapesPlot ---
polygons <- data.frame(
   id = paste0("poly_", 1:10),
   cat = sample(LETTERS[1:3], 10, replace = TRUE),
   feat1 = rnorm(10),
   feat2 = rnorm(10),
   geometry = c(
       'POLYGON((64.6 75.3,66.0 70.5,66.4 70.2,67.0 69.8,72.8 70.4,64.6 75.3))',
       'POLYGON((56.7 63.0,52.3 65.6,48.0 63.2,51.2 55.7,57.1 59.2,56.7 63.0))',
       'POLYGON((9.9 16.5,9.3 15.9,8.0 13.1,11.5 7.8,17.8 11.3,9.9 16.5))',
       'POLYGON((64.9 37.2,60.3 37.4,57.6 31.7,58.9 29.3,64.0 28.1,64.9 37.2))',
       'POLYGON((30.5 49.1,22.4 46.5,22.4 43.9,30.9 41.9,31.6 42.9,30.5 49.1))',
       'POLYGON((78.3 57.8,70.5 61.6,71.6 52.7,72.2 52.5,77.4 54.5,78.3 57.8))',
       'POLYGON((41.8 23.8,41.3 25.9,41.0 26.4,36.5 28.7,35.8 28.6,41.8 23.8))',
       'POLYGON((15.7 75.9,14.2 74.4,15.7 67.5,23.0 69.8,23.4 71.7,15.7 75.9))',
       'POLYGON((80.7 37.4,75.3 31.3,77.1 28.5,82.5 28.0,83.1 28.5,80.7 37.4))',
       'POLYGON((15.5 37.8,14.4 38.6,7.3 32.6,8.3 30.9,15.1 30.2,15.5 37.8))'
   )
)

polygons <- terra::vect(polygons, crs = "EPSG:4326", geom = "geometry")

SpatShapesPlot(polygons)
SpatShapesPlot(polygons, ext = c(0, 20, 0, 20))
SpatShapesPlot(polygons, highlight = 'cat == "A"', highlight_color = "red2")
SpatShapesPlot(polygons, border_color = "red", border_size = 2)
SpatShapesPlot(polygons, fill_by = "cat", fill_name = "category")
# Let border color be determined by fill
SpatShapesPlot(polygons, fill_by = "cat", alpha = 0.6, border_color = TRUE)
SpatShapesPlot(polygons, fill_by = "feat1")
SpatShapesPlot(polygons, fill_by = c("feat1", "feat2"), palette = "RdYlBu")

# --- SpatPointsPlot ---
# create some random points in the above polygons
points <- data.frame(
  id = paste0("point_", 1:30),
  gene = sample(LETTERS[1:3], 30, replace = TRUE),
  feat1 = runif(30, 0, 100),
  feat2 = runif(30, 0, 100),
  size = runif(30, 1, 5),
```

```
  x = c(
    61.6, 14.3, 12.7, 49.6, 74.9, 58.9, 13.9, 24.7, 16.9, 15.6,
    72.4, 60.1, 75.4, 14.9, 80.3, 78.8, 16.7, 27.6, 48.9, 52.5,
    12.9, 11.8, 50.4, 25.6, 10.4, 51.9, 73.4, 26.8, 50.4, 60.0
  ),
  y = c(
    32.1, 12.8, 33.2, 59.9, 57.8, 31.9, 10.1, 46.8, 75.3, 69.0,
    60.0, 29.4, 54.2, 34.2, 35.3, 33.1, 74.7, 48.0, 63.2, 59.2,
    9.2, 15.1, 64.5, 47.1, 11.4, 60.1, 54.1, 44.5, 61.9, 30.3
  )
)

SpatPointsPlot(points)
SpatPointsPlot(points, color_by = "gene", size_by = "size", shape = 22,
  border_size = 1)
SpatPointsPlot(points, raster = TRUE, raster_dpi = 30, color_by = "feat1")
SpatPointsPlot(points, color_by = c("feat1", "feat2"), size_by = "size")
SpatPointsPlot(points, color_by = "feat1", upper_cutoff = 50)
SpatPointsPlot(points, color_by = "feat1", hex = TRUE)
SpatPointsPlot(points, color_by = "gene", label = TRUE)
SpatPointsPlot(points, color_by = "gene", highlight = 1:20,
  highlight_color = "red2", highlight_stroke = 0.8)

# --- Graph/Network functionality ---
# Create a simple adjacency matrix for demonstration
set.seed(8525)
graph_mat <- matrix(0, nrow = 30, ncol = 30)
# Add some random connections with weights
for(i in 1:30) {
  neighbors <- sample(setdiff(1:30, i), size = sample(2:5, 1))
  graph_mat[i, neighbors] <- runif(length(neighbors), 0.1, 1)
}
rownames(graph_mat) <- colnames(graph_mat) <- rownames(points)
attr(points, "graph") <- graph_mat

SpatPointsPlot(points, color_by = "gene", graph = "@graph",
  edge_color = "grey60", edge_alpha = 0.3)
SpatPointsPlot(points, color_by = "feat1", graph = graph_mat,
  edge_size = c(0.1, 1), edge_alpha = 0.5)

# --- Use the `return_layer` argument to get the ggplot layers
ext = c(0, 40, 0, 50)
ggplot2::ggplot() +
  SpatImagePlot(r, return_layer = TRUE, alpha = 0.2, ext = ext) +
  SpatShapesPlot(polygons, return_layer = TRUE, ext = ext, fill_by = "white") +
  SpatPointsPlot(points, return_layer = TRUE, ext = ext, color_by = "feat1") +
  theme_box() +
  ggplot2::coord_sf(expand = 0) +
  ggplot2::scale_y_continuous(labels = function(x) -x)
```

theme_blank *Blank theme*

Description

This function creates a theme with all elements blank except for axis lines and labels. It can optionally add coordinate axes in the plot.

Usage

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

Arguments

add_coord	Whether to add coordinate arrows. Default is TRUE.
xlen_npc	The length of the x-axis arrow in "npc".
ylen_npc	The length of the y-axis arrow in "npc".
xlab	x-axis label.
ylab	y-axis label.
lab_size	Label size.
...	Arguments passed to the theme.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
    geom_point()
p + theme_blank()
p + theme_blank(xlab = "x-axis", ylab = "y-axis", lab_size = 16)
```

theme_box *Box theme*

Description

This function creates a theme with all elements blank except for axis lines like a box around the plot.

Usage

```
theme_box(
  xlen_npc = 0.15,
  ylen_npc = 0.15,
  xlab = "",
  ylab = "",
  lab_size = 12,
  ...
)
```

Arguments

xlen_npc	The length of the x-axis arrow in "npc".
ylen_npc	The length of the y-axis arrow in "npc".
xlab	x-axis label.
ylab	y-axis label.
lab_size	Label size.
...	Arguments passed to the theme.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
    geom_point()
p + theme_box()
```

theme_this *A ggplot2 theme and palettes for plotthis Borrowed from the* theme_this *function in the* SCP *pipeline*

Description

A ggplot2 theme and palettes for plotthis Borrowed from the theme_this function in the SCP pipeline

Usage

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

Arguments

aspect.ratio	The aspect ratio of the plot
base_size	The base size of the text If not specified, it will use the value from getOption("theme_this.base_size" 12). If you want to change the default base size, you can set the option theme_this.base_size. This is applied to all plots using this theme.
font_family	The font family of the text If not specified, it will use the value from getOption("theme_this.font_fam If you want to change the default font family, you can set the option theme_this.font_family. This is applied to all plots using this theme. To list available font families, you can use the systemfonts::system_fonts() function.
...	Other arguments for theme()

Value

A ggplot2 theme

See Also

https://github.com/zhanghao-njmu/SCP

TrendPlot *Trend plot*

Description

Draws a trend plot combining stacked bars with a semi-transparent area background to show how one or more groups accumulate across a discrete x-axis variable. This hybrid style sits between a bar plot and an area plot: it preserves the discrete category separation of bars while softening the visual with an area fill, making trends across categories easier to perceive.

The function supports **count aggregation** (omit y to plot observation counts per x-category), **proportion scaling** (scale_y = TRUE normalises each x position to 100\ colour control, faceting, and splitting into separate sub-plots via split_by.

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_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  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 = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.

x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string used to join multiple x columns. Default "_". Ignored when x is a single column.
split_by	The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string used as the fill legend title. When NULL, the group_by column name is used.
scale_y	A logical value. When TRUE, y-values are scaled to proportions within each (x, facet_by) group so that each x position stacks to 1.0. The y-axis labels switch from numeric to percent format automatically.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
x_text_angle	A numeric value specifying the angle of the x-axis text.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

seed
: A numeric seed for reproducibility. Passed to `validate_common_args()`.

combine
: Logical; when `TRUE` (default), returns a combined `patchwork` object. When `FALSE`, returns a named list of individual `ggplot` objects.

ncol, nrow
: Integer number of columns / rows for the combined layout (passed to `wrap_plots`).

byrow
: Logical; fill the combined layout by row. Default `TRUE` (passed to `wrap_plots`).

axes
: A character string specifying how axes should be treated across the combined layout (passed to `wrap_plots`).

axis_titles
: A character string specifying how axis titles should be treated across the combined layout. Defaults to `axes`.

guides
: A character string specifying how guides (legends) should be collected across panels. Default `"collect"` (passed to `combine_plots()`).

design
: A custom layout design for the combined plot (passed to `combine_plots()`).

...
: Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. check_keep_na() and check_keep_empty() normalise the keep_na / keep_empty arguments for all columns (x, split_by, group_by, facet_by).
2. The split_by column is validated and its NA / empty levels are processed via process_keep_na_empty(). It is then removed from the per-column keep_na / keep_empty lists.
3. The data frame is split by split_by (preserving level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
5. TrendPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
6. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

See Also

AreaPlot for a pure stacked area plot.

Examples

```
data <- data.frame(
    x = factor(rep(c("A", "B", NA, "D"), 3), levels = LETTERS[1:4]),
    y = c(1, 3, 6, 5, 4, 2, 5, 7, 8, 9, 4, 8),
    group = factor(rep(c("F1", NA, "F3"), each = 4), levels = c("F1", "F2", "F3"))
)

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

# Scaled to proportions
TrendPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)

# Split into sub-plots (no group_by -- single-colour fill)
TrendPlot(data, x = "x", y = "y", split_by = "group")

# Per-split palettes
TrendPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(F1 = "Set1", F3 = "Dark2"))

# How keep_na and keep_empty work
TrendPlot(data, x = "x", y = "y", group_by = "group",
```

```
         keep_na = TRUE, keep_empty = TRUE)
TrendPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))

# Faceting
TrendPlot(data, x = "x", y = "y", facet_by = "group",
         keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))
```

`UpsetPlot` *UpSet Plot*

Description

Draws an UpSet plot visualising set intersections and set sizes. The plot comprises:

- A **horizontal bar chart** showing the size of each intersection, filled by the intersection count.
- A **combination matrix** (rows = sets, columns = intersections) with membership dots and connecting lines.
- A **set-size bar chart** on the left of the matrix (added automatically by `ggupset`).

The function accepts data in four formats:

- **List** — a named list of element vectors (one per set).
- **Long** — a data frame with one row per (set, element) pair.
- **Wide** — a data frame where each row is an element and each set has its own logical or 0/1 membership column.
- **UpsetPlotData** — a pre-processed object from `prepare_upset_data()`.

Supports splitting into sub-plots via `split_by`, per-split colour palettes and legend control, and combining sub-plots via `patchwork`.

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 = "Blues",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  specific = TRUE,
  theme = "theme_this",
  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,
  combmatrix_gap = 6,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
in_form	A character string specifying the input format. One of "auto" (default; detect from data structure), "long", "wide", "list", or "upset".
split_by	The column(s) to split the data by and produce separate sub-plots. Only supported for data.frame input (list input raises an error). Multiple columns concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default: "_".
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
id_by	A character string specifying the column name for instance identifiers. Required for long format; optional for wide format (a synthetic .id column is created if omitted).
label	A logical value. When TRUE (default), count labels are displayed above each intersection bar via geom_text_repel().
label_fg	A character string specifying the colour of the label text. Default: "black".

label_size A numeric value specifying the size of the label text. Default: NULL (computed from base_size / 12 * 3.5).

label_bg A character string specifying the background fill colour of the label. Default: "white".

label_bg_r A numeric value specifying the corner radius of the label background, passed to geom_text_repel(bg.r). Default: 0.1.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

alpha A numeric value specifying the transparency of the plot.

specific A logical value. When TRUE (default), only specific intersections are returned (elements belonging exclusively to the shown set combination). When FALSE, all overlapping items are included. See https://github.com/gaospecial/ggVennDiagram/issues/64.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

combine Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

seed A numeric seed for reproducibility. Passed to validate_common_args(). Default: 8525.

combmatrix_gap A numeric value specifying the gap between rows of the combination matrix, measured at base_size = 12. The actual gap is scaled by text_size_scale = base_size / 12. Default: 6.

axes	A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).
axis_titles	A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.
guides	A character string specifying how legends should be collected across panels (passed to combine_plots()).
design	A custom layout design for the combined plot (passed to combine_plots()).
...	Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. **Guard** — split_by is only supported for data.frame input. If data is a list (or other non-data.frame type) and split_by is non-NULL, an error is raised.
2. **Column validation** — check_columns() resolves the split_by column(s) with force_factor = TRUE and allow_multi = TRUE. Multiple columns are concatenated with split_by_sep.
3. **Data splitting** — empty factor levels in split_by are dropped via droplevels(), then the data frame is split by split_by (level order is preserved). If split_by is NULL, the data is wrapped in a single-element list with name "...".
4. **Per-split resolution** — check_palette(), check_palcolor(), and check_legend() resolve per-split palette, palcolor, legend.position, and legend.direction.
5. **Atomic dispatch** — UpsetPlotAtomic is called for each split. If title is a function, it receives the split level name and can generate dynamic titles. When in wide mode (in_form is "auto" or "wide") and group_by is NULL, the set columns are auto-detected as all columns except id_by and split_by.
6. **Combination** — Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
# ---- list input ---------------------------------------------------------
data <- list(
    A = 1:5,
    B = 2:6,
    C = 3:7,
    D = 4:8
)
UpsetPlot(data)
UpsetPlot(data, label = FALSE)
UpsetPlot(data, palette = "Reds", specific = FALSE)

# ---- long-format data frame ---------------------------------------------
```

```
data_long <- data.frame(
    group_by = factor(
         c(rep("A", 5), rep("B", 5), rep("C", 5), rep("D", 5)),
         levels = c("A", "B", "C", "D")
    ),
    id_by = c(1:5, 2:6, 3:7, 4:8)
)
UpsetPlot(data_long, in_form = "long", group_by = "group_by", id_by = "id_by")

# ---- wide-format data frame ------------------------------------------
data <- data.frame(
    id = LETTERS[1:10],
    B = c(1, 0, 1, 1, 0, 0, 1, 0, 1, 0),
    A = c(1, 1, 1, 0, 0, 1, 0, 0, 1, 0),
    D = c(1, 0, 0, 1, 1, 0, 0, 1, 0, 1),
    C = c(0, 1, 1, 0, 1, 0, 1, 0, 1, 0)
)
UpsetPlot(data, in_form = "wide", id_by = "id", n_intersections = 4)
```

`VelocityPlot` *Cell velocity plot*

Description

Plots RNA velocity vectors on a low-dimensional embedding (e.g., UMAP, t-SNE) to visualize the direction and magnitude of cellular state transitions. Supports three visualization modes: raw arrows at each cell position, arrows on a regular grid, and streamline paths. Optionally colors arrows by cell metadata groups.

Usage

```
VelocityPlot(
  embedding,
  v_embedding,
  plot_type = c("raw", "grid", "stream"),
  split_by = NULL,
  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,
  keep_na = FALSE,
  keep_empty = FALSE,
  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,
  palreverse = FALSE,
  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_this",
  theme_args = list(),
  return_layer = FALSE,
  seed = 8525
)
```

Arguments

embedding	A matrix or data frame of dimension n_obs x n_dim specifying the low-dimensional embedding coordinates (e.g., UMAP, t-SNE) of the cells. The first two columns are used for the x and y axes.
v_embedding	A matrix or data frame of dimension n_obs x n_dim specifying the velocity vectors for each cell. Must have the same dimensions as `embedding`.
plot_type	A character string specifying the visualization method. `"raw"` plots arrows directly from each cell's embedding position. `"grid"` averages velocities onto a regular grid and plots arrows at grid points. `"stream"` computes smooth streamline paths from the gridded velocity field. Default is `"raw"`.
split_by	Not supported for VelocityPlot. Setting this parameter will raise an error.
group_by	An optional vector of the same length as the number of rows in `embedding` specifying a grouping variable for cells. When provided, arrows are colored by group using `group_palette`. Only applies to `plot_type = "raw"`; ignored with a warning for `"grid"` and `"stream"`. Default is `NULL`.
group_name	A character string specifying the legend title for the grouping variable. Default is `"Group"`.

group_palette A character string specifying the color palette to use for the grouping variable. Passed to palette_this. Default is "Paired".

group_palcolor An optional character vector of specific colors for the grouping variable. If NULL, colors are generated from group_palette. Default is NULL.

n_neighbors An integer value specifying the number of nearest neighbors for computing grid velocities. Only used when plot_type is "grid" or "stream". Default is ceiling(nrow(embedding) / 50).

density A numeric value specifying the grid density along each dimension. Only used when plot_type is "grid" or "stream". For plot_type = "raw", when density is between 0 and 1, it specifies the fraction of cells to randomly subsample. Default is 1.

smooth A numeric value specifying the standard deviation multiplier for the Gaussian kernel when averaging cell velocities onto grid points. Only used when plot_type is "grid" or "stream". Default is 0.5.

scale A numeric value specifying the scaling factor for the velocity vectors. Applied to raw and grid arrows. For plot_type = "stream", this is fixed to 1 internally. Default is 1.

min_mass A numeric value specifying the minimum mass threshold for retaining grid points. Only used when plot_type is "grid" or "stream". Default is 1.

cutoff_perc A numeric value specifying the percentile cutoff for removing low-density grid points. Only used when plot_type is "stream". Default is 5.

arrow_angle A numeric value specifying the angle of the arrowheads in degrees. Applied to arrow when plot_type is "raw" or "grid". Default is 20.

arrow_color A character string specifying the color of the velocity arrows. For plot_type = "stream", this sets only the arrowhead color. Default is "black".

arrow_alpha A numeric value between 0 and 1 specifying the transparency of the velocity arrows. Only used when plot_type = "raw" or "grid"; for plot_type = "stream", use streamline_alpha instead. Default is 1.

keep_na A logical or character value specifying how to handle NA values in group_by. Unlike other plot functions, VelocityPlot does not support named lists for per-column control. See keep_na in common_args for details of supported values. Default is FALSE.

keep_empty One of FALSE, TRUE, or "level" specifying how to handle empty factor levels in group_by. Unlike other plot functions, VelocityPlot does not support named lists for per-column control. See keep_empty in common_args for details. Default is FALSE.

streamline_l A numeric value specifying the integration length of the streamlines. Passed to geom_streamline as the L parameter. Default is 5.

streamline_minl A numeric value specifying the minimum streamline length. Shorter streamlines are not drawn. Passed to geom_streamline as the min.L parameter. Default is 1.

streamline_res A numeric value specifying the resolution of the streamline integration. Passed to geom_streamline as the res parameter. Default is 1.

streamline_n A numeric value specifying the number of streamlines to draw. Passed to geom_streamline as the n parameter. Default is 15.

streamline_width
A numeric vector of length 2 specifying the range of line widths for streamlines. Passed to scale_size(range = ...). Only used when streamline_color is NULL. Default is c(0, 0.8).

streamline_alpha
A numeric value between 0 and 1 specifying the transparency of the velocity streamlines. Default is 1.

streamline_color
An optional character string specifying a fixed color for streamlines. When NULL (the default), streamlines are colored by velocity magnitude using streamline_palette.

streamline_palette
A character string specifying the color palette for streamline velocity magnitude. Passed to palette_this. Only used when streamline_color is NULL. Default is "RdYlBu".

streamline_palcolor
An optional character vector of specific colors for the streamline velocity gradient. If NULL, colors are generated from streamline_palette. Default is NULL.

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

streamline_bg_color
A character string specifying the background (outline) color applied to streamlines to create a stroke effect. Default is "white".

streamline_bg_stroke
A numeric value specifying the additional line width of the background stroke relative to the foreground streamline. Default is 0.5.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

return_layer A logical value indicating whether to return only the ggplot layers instead of the full assembled plot. When TRUE, returns a list of ggplot layers suitable for combining with other ggplot objects. Default is FALSE.

seed The random seed to use. Default is 8525.

Value

A ggplot object representing the cell velocity plot, with `height` and `width` attributes set for consistent rendering. If `return_layer = TRUE`, returns a list of ggplot layers instead.

Rendering Pipeline

The `VelocityPlot` function proceeds through the following steps:

1. **Input validation** — Verifies that `embedding` and `v_embedding` are matrices or data frames of equal dimensions, that `group_by` matches the number of rows (if provided), and that `split_by` is `NULL` (unsupported and raises an error).
2. **Axis label resolution** — Uses the column names of `embedding` as axis labels, falling back to `"Reduction 1"` and `"Reduction 2"` when column names are `NULL`.
3. **Grouping setup** — Converts `group_by` to a factor and applies `keep_na` / `keep_empty` logic to filter or recode missing values and empty factor levels.
4. **Plot-type dispatch** — Branches on `plot_type`:
   - **raw** — Optionally subsamples cells when `density < 1`, scales velocity vectors by `scale`, computes arrow lengths proportional to the embedding range, and renders `geom_segment` with arrowheads. When `group_by` is provided, arrows are colored by group using `group_palette`.
   - **grid** — Delegates to `.compute_velocity_on_grid` to interpolate the sparse cell velocities onto a regular grid, then renders `geom_segment` with arrowheads at each grid point. `group_by` is ignored with a warning.
   - **stream** — Delegates to `.compute_velocity_on_grid` with `adjust_for_stream = TRUE`, then renders smooth streamline paths via `geom_streamline`. When `streamline_color` is provided, streamlines use a fixed color with a background stroke; when `NULL`, streamlines are colored by velocity magnitude using `streamline_palette`. `group_by` is ignored with a warning.
5. **Layer return or plot assembly** — If `return_layer = TRUE`, returns the list of ggplot layers. Otherwise, constructs a full `ggplot` object with labels, theme, aspect ratio, legend configuration, and `height` / `width` attributes via `calculate_plot_dimensions()`.

See Also

`DimPlot` `FeatureDimPlot`

Examples

```
data(dim_example)
dim_example$clusters[dim_example$clusters == "Ductal"] <- NA

# Basic velocity plot with group coloring
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters)

# Handle NA groups with keep_na / keep_empty
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = TRUE)
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = 'level')
```

```
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = FALSE)
```

VennDiagram *Venn / Euler diagram*

Description

Draws Venn or Euler diagrams that visualise the overlap relationships among multiple sets. Supports four input formats: long (one row per element-set pair), wide (logical/0-1 columns per set), a named list (element vectors per set), and a pre-computed `VennPlotData` object.

Intersection regions can be filled by a continuous colour gradient encoding the element count (`fill_mode = "count"` / `"count_rev"`) or by blended set colours (`fill_mode = "set"`). Region labels can display counts, percentages, both, or a custom function. Set labels always show the set name and its total element count.

Use `split_by` to produce separate Venn diagrams for each level of a grouping variable. Note that `split_by` is only supported when `data` is a data frame (list and `VennPlotData` inputs cannot be split).

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",
  palreverse = FALSE,
  fill_name = NULL,
  palette = ifelse(fill_mode == "set", "Paired", "Blues"),
  palcolor = NULL,
  alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  aspect.ratio = 1,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
in_form	A character string specifying the input format. One of "auto" (default; detect automatically via detect_venn_datatype()), "long", "wide", "list", or "venn".
split_by	The column(s) to split the data by and produce separate Venn diagrams per subgroup. Only supported when data is a data frame. Multiple columns are concatenated with split_by_sep.
split_by_sep	A character string to separate concatenated split_by columns. Default "_".
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
id_by	A character string specifying the column name that identifies individual elements. Required for long-format data; ignored otherwise.
label	A character string or function controlling the text shown in each intersection region. One of: • "count" (default) — the raw count of elements in that region. • "percent" — the percentage of the total element count. • "both" — count and percentage on separate lines. • "none" — no region labels are drawn. • A **function** — receives a data frame with columns "id", "X", "Y", "name", "item", and "count", and must return a character vector of labels.
label_fg	A character string specifying the colour of the label text.
label_size	A numeric value specifying the font size of the label text. When NULL (the default), auto-sized at 3.5 for region labels and 4 for set labels, scaled by base_size / 12.
label_bg	A character string specifying the background colour of the label text (passed to geom_text_repel() as bg.color). Default "white".
label_bg_r	A numeric value specifying the corner radius of the label background rectangle (passed as bg.r). Default 0.1.

fill_mode
: A character string specifying how intersection regions are coloured. One of:
  - "count" — continuous gradient based on element count (default palette: "Spectral").
  - "count_rev" — continuous gradient with reversed count order.
  - "set" — discrete blended colours per set combination (default palette: "Paired"). No legend is drawn.

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

fill_name
: A character string for the colour bar legend title when fill_mode is "count" or "count_rev". Ignored when fill_mode = "set".

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha
: A numeric value specifying the transparency of the plot.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

combine
: Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow
: Integer number of columns / rows for the combined layout when combine = TRUE.

byrow
: Logical; fill the combined layout by row (default TRUE).

seed
: A numeric seed for reproducibility. Default 8525.

axes, axis_titles
: Character strings specifying how axes and axis titles are handled across the combined layout.

guides
: A character string specifying how legends are collected across panels in the combined layout.

design
: A custom layout specification for the combined plot. Passed to wrap_plots().

...
: Additional arguments.

Value

A ggplot object (single split), a patchwork object (combined sub-plots), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When a non-NULL split_by is provided and the input is a data frame:

1. **Validation** — an error is raised if data is not a data frame (list and VennPlotData input cannot be split).
2. **Column resolution** — split_by is validated and optionally concatenated via check_columns() with force_factor = TRUE and allow_multi = TRUE.
3. **Data splitting** — the data frame is split by the unique levels of the split_by column, preserving factor level order. Empty levels are dropped via droplevels().
4. **Per-split colour and legend resolution** — check_palette(), check_palcolor(), and check_legend() resolve per-split palettes, custom colours, legend positions, and legend directions.
5. **Atomic dispatch** — VennDiagramAtomic() is called for each subset. When title is a function, it receives the split level name for dynamic title generation.
6. **Combination** — results are passed to combine_plots() which returns a combined patchwork object (when combine = TRUE) or a named list of individual ggplot objects (when combine = FALSE).

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))
)

# Basic Venn diagram with count labels
VennDiagram(data)

# Fill by set membership (blended colours)
VennDiagram(data, fill_mode = "set")

# Show both count and percentage
VennDiagram(data, label = "both")

# Custom label function using set names
VennDiagram(data, label = function(df) df$name)

# Custom palette and transparency
VennDiagram(data, palette = "material-indigo", alpha = 0.6)
```

VolcanoPlot *Volcano plot*

Description

Produces a volcano plot — a scatter plot that displays statistical significance (typically -log10 adjusted p-value) on the y-axis versus magnitude of change (log2 fold change) on the x-axis. Points are coloured automatically by significance category (`"sig_pos_x"`, `"sig_neg_x"`, `"insig"`) or by a user-supplied column. The most significant features can be labelled automatically via `geom_text_repel()`, and specific points can be highlighted.

The function supports **automatic labelling** of top features (by distance to origin), **mirrored layout** via `flip_negatives`, **x-axis trimming** to reduce the influence of extreme values, **faceting**, and **splitting** into separate sub-plots via `split_by` with per-split colour palette and legend control.

Usage

```
VolcanoPlot(
  data,
  x,
  y,
  ytrans = "-log10",
  color_by = NULL,
  color_name = NULL,
  xlim = NULL,
  flip_negatives = FALSE,
  x_cutoff = NULL,
  y_cutoff = 0.05,
  split_by = NULL,
  split_by_sep = "_",
  label_by = 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,
  pt_shape = 21,
  pt_border_color = TRUE,
  pt_border_size = 0.5,
  nlabel = 5,
  labels = 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,
  raster = NULL,
  raster_dpi = c(512, 512),
  trim = c(0, 1),
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  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 = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
ytrans	A function or a function name (as a string) to transform the y-axis values before

plotting. The transformed values are used for both the y-axis and cutoff comparisons. Default: "-log10" (converts p-values to a -log10 scale). Other named functions can be passed as strings, e.g. "sqrt".

color_by A character string specifying the column name to colour the points by. When NULL (default), points are automatically categorised as "sig_pos_x", "sig_neg_x", or "insig" based on x_cutoff and y_cutoff, and the colour legend is suppressed. When a column name is provided, the colour mapping follows the column type — discrete (character/factor) uses scale_color_manual() with the specified palette; numeric (continuous) uses scale_color_gradientn().

color_name A character string for the colour legend title when color_by is a numeric column. When NULL (default), the color_by column name is used.

xlim A numeric vector of length 2 to set the x-axis limits. Passed to xlim(). When NULL (default), limits are determined automatically from the data.

flip_negatives A logical value. When TRUE, y-values of points with negative x-values are multiplied by -1, creating a mirrored volcano plot where both up- and down-regulated features show their significance on the same side of the y-axis. A horizontal line at y = 0 and absolute-value axis labels are added. Default: FALSE.

x_cutoff A numeric value specifying the x-axis significance cutoff. Both the negative and positive of this value are used as vertical threshold lines. When NULL or 0, no x-cutoff line is drawn. Default: NULL.

y_cutoff A numeric value specifying the y-axis significance cutoff in the **original** (untransformed) scale. The value is transformed by ytrans before plotting. When NULL, no y-cutoff line is drawn and the category assignment uses only the x-cutoff. Default: 0.05.

split_by The column(s) to split the data by and produce separate sub-plots. Multiple columns are concatenated with split_by_sep.

split_by_sep A character string to separate concatenated split_by columns. Default "_".

label_by A character string specifying the column whose values are used as label text. When NULL (default), row names of the data frame are used.

x_cutoff_name A character string for the x-cutoff legend entry. When "none", the legend for the x-cutoff line is suppressed entirely (the line is still drawn). When NULL (default), a label of the form "<x> = +/-<value>" is generated.

y_cutoff_name A character string for the y-cutoff legend entry. When "none", the legend for the y-cutoff line is suppressed entirely (the line is still drawn). When NULL (default), a label of the form "<ylab> = <value>" is generated.

x_cutoff_color A character string specifying the colour of the x-axis cutoff line(s). Default: "red2".

y_cutoff_color A character string specifying the colour of the y-axis cutoff line(s). Default: "blue2".

x_cutoff_linetype
A character string specifying the linetype of the x-axis cutoff line(s). Default: "dashed".

y_cutoff_linetype
A character string specifying the linetype of the y-axis cutoff line(s). Default: "dashed".

x_cutoff_linewidth
: A numeric value specifying the linewidth of the x-axis cutoff line(s). Default: 0.5.

y_cutoff_linewidth
: A numeric value specifying the linewidth of the y-axis cutoff line(s). Default: 0.5.

pt_size
: A numeric value specifying the point size for all data points. Default: 2.

pt_alpha
: A numeric value in [0, 1] specifying the transparency of all data points. Default: 0.5.

pt_shape
: A numeric value specifying the point shape. Default: 21 (filled circle with border). Shapes 21–25 support separate fill and border colour aesthetics; all other shapes use a single colour aesthetic. In raster mode, all points are drawn as filled circles (shape is ignored).

pt_border_color
: Controls the point border colour. For shapes 21–25:

  - TRUE (default) – border colour tracks the color_by gradient / palette.
  - A colour string (e.g. "black") – constant colour border.

  FALSE or NULL disables the border. For shapes without a fill aesthetic (not 21–25), this parameter has no effect. In raster mode the border is drawn as a slightly larger disc behind each point (the shape is always a circle there), and TRUE falls back to a border disc in the color_by colour.

pt_border_size
: A numeric value specifying the point border size (stroke width, in mm). 0 disables the border. Default: 0.5.

nlabel
: An integer specifying the number of top features to label automatically. Points are ranked by Euclidean distance to the origin within each sign(x) group (and per facet level if facet_by is set). Only non-insignificant points receive labels. Default: 5.

labels
: A character vector of row names or integer indices specifying which points to label. Overrides automatic nlabel selection. When NULL (default), top nlabel points are chosen automatically.

label_size
: A numeric value specifying the font size of the labels. Default: 3.

label_fg
: A character string specifying the text colour of the labels. Default: "black".

label_bg
: A character string specifying the background colour of the label boxes (passed to geom_text_repel(bg.color = ...)). Default: "white".

label_bg_r
: A numeric value specifying the corner radius of the label background boxes (passed to geom_text_repel(bg.r = ...)). Default: 0.1.

highlight
: A character vector of row names or integer indices specifying which points to highlight with an overlaid point layer in highlight_color. When NULL (default), no highlighting is applied.

highlight_color
: A character string specifying the colour of the highlight points. Default: "red".

highlight_size
: A numeric value specifying the point size of the highlight layer. Default: 2.

highlight_alpha
: A numeric value in [0, 1] specifying the transparency of the highlight points. Default: 1.

highlight_stroke
: A numeric value specifying the stroke width of the highlight point borders. Default: 0.5.

raster
: A logical value. If TRUE, points are rendered via scattermore::geom_scattermore() for efficient rasterised plotting. Default is NULL, which auto-enables when nrow(data) > 1e3.

raster_dpi
: A numeric vector of length 2 [x_dpi, y_dpi] specifying the raster resolution in pixels. Passed to scattermore::geom_scattermore(pixels = ...). Default is c(512, 512). If a single value is provided it is recycled to both dimensions.

trim
: A numeric vector of length 2 specifying quantile bounds for winsorizing the x-axis values. Values below the first quantile are clamped to that quantile; values above the second quantile are clamped to that quantile. Both values must be in [0, 1]. When both bounds are nonzero and of opposite sign, they are symmetrised to the smaller absolute value. Default: c(0, 1) (no trimming).

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Logical; when TRUE (default), returns a combined patchwork object. When FALSE, returns a named list of individual ggplot objects.

ncol, nrow
: Integer number of columns / rows for the combined layout (passed to wrap_plots).

byrow
: Logical; fill the combined layout by row. Default TRUE (passed to wrap_plots).

axes
: A character string specifying how axes should be treated across the combined layout (passed to wrap_plots).

axis_titles
: A character string specifying how axis titles should be treated across the combined layout. Defaults to axes.

guides
: A character string specifying how guides (legends) should be collected across panels. Default "collect" (passed to combine_plots()).

design
: A custom layout design for the combined plot (passed to combine_plots()).

...
: Additional arguments.

Value

A ggplot object, a patchwork object, or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by Workflow

When split_by is provided:

1. The split_by column(s) are validated via check_columns() with force_factor = TRUE and concat_multi = TRUE (multiple columns are concatenated with split_by_sep).
2. The data frame is split by split_by (preserving factor level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
3. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
4. VolcanoPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles.
5. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list.

Examples

```
set.seed(8525)
n <- 200
n_de <- 150

## Non-DE genes
fc_null <- rnorm(n - n_de, 0, 0.35)
z_null  <- rnorm(n - n_de, 0, 1)
```

```
## DE genes
fc_de <- rnorm(
  n_de,
  mean = sample(c(-1, 1), n_de, replace = TRUE),
  sd = 0.45
)

## Make significance related to effect size,
## but with substantial variation
z_de <- fc_de * rnorm(n_de, 4.5, 0.8)

avg_log2FC <- c(fc_null, fc_de)
z <- c(z_null, z_de)

p_val <- 2 * pnorm(-abs(z))
p_val_adj <- p.adjust(p_val, method = "BH")

## Shuffle genes
i <- sample(n)

data <- data.frame(
  avg_log2FC = avg_log2FC[i],
  p_val_adj = p_val_adj[i],
  gene = paste0("gene", seq_len(n))[i],
  pct_diff = rnorm(n, 0, 1),
  group = sample(LETTERS[1:2], n, replace = TRUE)
)

# --- Basic usage ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)")
# --- With gene labels ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", label_by = "gene")
# --- Mirrored layout ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, label_by = "gene")
# --- With faceting ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, facet_by = "group", label_by = "gene")
# --- With splitting ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, split_by = "group", label_by = "gene")
# --- With highlighting ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   highlight = c("gene196", "gene151"), label_by = "gene")
# --- Per-split palettes ---
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", split_by = "group", label_by = "gene",
   palette = c(A = "Spectral", B = "PuOr"))
# Trim extreme x-values (winsorize to 40% and 50% quantiles), for demo purposes
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", trim = c(0.4, 0.5))
```

WordCloudPlot *Word cloud plot*

Description

Draws a word cloud plot that visualises word frequency and importance. Words are displayed with font size proportional to a count variable and colour based on a continuous score variable, using geom_text_wordcloud for rendering.

The function supports **pre-tokenised words** (via word_by) and **sentence splitting** (via sentence_by). Sentences are automatically lowercased, stripped of punctuation, and split into individual words before aggregation. Common stop words can be excluded via words_excluded, and the number of displayed words is controlled by top_words.

The word cloud can be **faceted** (via facet_by) or **split** into separate sub-plots via split_by. When split_by is used, each split level receives its own word cloud, and the results are combined into a single layout via combine_plots.

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 = plotthis::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_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  palreverse = FALSE,
```

```
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = 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.
word_by	A character string specifying the column name containing pre-tokenized words. A character column is expected. Use this when your data already has one word per row (or a list of words per row). Mutually exclusive with sentence_by.
sentence_by	A character string specifying the column name containing sentences or phrases to be split into individual words. A character column is expected. The text is lowercased and punctuation is removed before splitting on whitespace boundaries. Mutually exclusive with word_by.
count_by	A character string specifying the numeric column for the count or frequency of each word. When NULL (the default), each occurrence counts as 1. Must be NULL when sentence_by is used, as counts are derived from the number of occurrences after splitting.
score_by	A character string specifying the numeric column for the score of each word, mapped to the text colour via a continuous gradient. When NULL (the default), all words receive a score of 1 and are coloured at the low end of the palette.
count_name	A character string for the size legend title. When NULL (the default), "Count" is used.
score_name	A character string for the colour-bar legend title. When NULL (the default), "Score" is used.
split_by	The column(s) to split data by and plot separately. When combine = TRUE, the combined plot's data (p$data) contains the rows of every split, tagged with the split_by column. For ggraph-based plots (e.g. Network, ClustreePlot), the node layout of every split is exposed with the split column, and the original link rows (also tagged with the split_by column) are available as the "edges" attribute of the data.
split_by_sep	The separator for multiple split_by columns. See split_by

words_excluded A character vector of words to exclude from the word cloud. Matching is case-insensitive. Defaults to plotthis::words_excluded, a built-in set of common English stop words.

score_agg A function to aggregate the scores when multiple observations of the same word exist. Default is mean. Other options include sum, median, or a custom function.

minchar A numeric value specifying the minimum number of characters a word must have to be included. Words with fewer characters are filtered out. Default: 2.

word_size A numeric vector of length 2 specifying the range of font sizes (in mm) for the words. Passed to scale_size(range = word_size). Default: c(2, 8).

top_words A numeric value specifying the maximum number of words to display, selected by highest score. Default: 100.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha A numeric value specifying the transparency of the plot.

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
axes	A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axes in individual plots. • 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout. • 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.
axis_titles	A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axis titles in individual plots. • 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction. • 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.
guides	A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'collect' will collect guides below to the given nesting level, removing duplicates. • 'keep' will stop collection at this level and let guides be placed alongside their plot. • 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.
design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.

Value

A ggplot object (when no split_by is used), a patchwork object (when combine = TRUE and split_by is used), or a named list of ggplot objects (when combine = FALSE), each with height and width attributes in inches.

split_by workflow

When split_by is provided:

1. validate_common_args() validates the seed and facet_by constraints (maximum 2 facet columns).

2. The theme argument is resolved via process_theme().
3. The split_by column(s) are validated and transformed via check_columns() with force_factor = TRUE and concat_multi = TRUE.
4. The data frame is split by split_by (preserving factor level order). If split_by is NULL, the data is wrapped in a single-element list with name "...".
5. Per-split palette, palcolor, legend.position, and legend.direction are resolved via check_palette(), check_palcolor(), and check_legend().
6. WordCloudPlotAtomic() is called for each split. If title is a function, it receives the split level name and can generate dynamic titles per sub-plot.
7. Results are combined via combine_plots() (when combine = TRUE) or returned as a named list (when combine = FALSE).

Examples

```
set.seed(8525)
data <- data.frame(
    word = c("apple", "banana", "cherry", "date", "elderberry",
             "fig", "grape", "honeydew", "kiwi", "lemon"),
    count = c(10, 20, 30, 40, 50, 15, 25, 35, 45, 55),
    score = c(1, 2, 3, 4, 5, 1.5, 2.5, 3.5, 4.5, 5.5),
    facet = rep(c("Group1", "Group2"), each = 5),
    split = rep(c("A", "B"), 5)
)

# Basic word cloud with word, count, and score columns
WordCloudPlot(data, word_by = "word",
              count_by = "count", score_by = "score")

# Word cloud using sentence_by (sentences split into words)
data_sent <- data.frame(
    sentence = c("The quick brown fox jumps over the lazy dog",
                 "A quick brown dog jumps over a lazy fox"),
    score = c(10, 5)
)
WordCloudPlot(data_sent, sentence_by = "sentence", score_by = "score")

# Word cloud with faceting
WordCloudPlot(data, word_by = "word",
              count_by = "count", score_by = "score",
              facet_by = "facet")

# Word cloud split by a grouping variable
WordCloudPlot(data, word_by = "word",
              count_by = "count", score_by = "score",
              split_by = "split")
```

words_excluded *Excluded words in keyword enrichment analysis and extraction*

Description

The variable "words_excluded" represents the words that are excluded during keyword enrichment analysis or keyword extraction process. These mainly include words that are excessively redundant or of little value.

Examples

```
## Not run:
if (interactive()) {
  words_excluded <- c(
    "the", "is", "and", "or", "a", "in", "on", "under", "between", "of", "through",
    "via", "along", "that", "for", "with", "within", "without", "cell", "cellular",
    "dna", "rna", "protein", "peptide", "amino", "acid", "development", "involved",
    "organization", "system", "regulation", "regulated", "positive", "negative",
    "response", "process", "processing", "small", "large", "change", "disease"
  )
}

## End(Not run)
```

Index

∗ **data**
    dim_example, 77
    enrich_example, 95
    enrich_multidb_example, 96
    gsea_example, 101
    palette_list, 158
    words_excluded, 262
∗ **spatial**
    SpatImagePlot, 217
.compute_velocity_on_grid, *245*
.flip_y, *226, 227*
.wrap_spatial_layers, *226, 227*

adjust_network_layout, *94*
AlluvialPlot *(*SankeyPlot*)*, 202
anno_boxplot, *111*
anno_density, *111*
anno_lines, *111*
anno_pie, *111*
anno_points, *111*
anno_simple, *111*
anno_violin, *111*
AreaPlot, 3, *236*
AreaPlotAtomic, *6*
arrange, *122*
arrow, *71*, *155*, *243*

BarPlot, 8
BarPlotAtomic, *17*
BeeswarmPlot *(*BoxPlot*)*, 20
bg_layer, *82*, *123*, *128*
BoxPlot, 20, *45*
BoxViolinPlot, *20, 21*
BoxViolinPlotAtomic, *20*

calculate_plot_dimensions, *245*
check_columns, *47*, *52*, *94*, *125*, *131*, *147*, *156*, *172*, *183*, *200*, *209*, *215*, *240*, *249*, *255*, *261*
check_keep_empty, *6*, *16, 17*, *38*, *125*, *166*, *178*, *193*, *236*
check_keep_na, *6*, *16, 17*, *38*, *125*, *166*, *178*, *193*, *236*
check_legend, *6*, *16*, *47*, *52*, *94*, *125*, *131*, *167*, *172*, *178*, *183*, *193*, *200*, *215*, *236*, *240*, *249*, *255*, *261*
check_palcolor, *6*, *16*, *39*, *47*, *52*, *94*, *125*, *131*, *147*, *167*, *172*, *178*, *183*, *193*, *200*, *215*, *236*, *240*, *249*, *255*, *261*
check_palette, *6*, *16*, *39*, *47*, *52*, *94*, *125*, *131*, *147*, *167*, *172*, *178*, *183*, *193*, *200*, *215*, *236*, *240*, *249*, *255*, *261*
ChordPlot, *35*, 35
ChordPlotAtomic, *39*
CircosPlot *(*ChordPlot*)*, 35
ClustreePlot, 39, *56*, *91*, *135*, *152*, *169*, *186*, *205*, *258*
ClustreePlotAtomic, *41*, *43*
combine_plots, *6*, *17*, *39*, *42, 43*, *47*, *52*, *84, 85*, *94*, *100*, *124, 125*, *131, 132*, *146, 147*, *156*, *166, 167*, *172*, *178, 179*, *182, 183*, *193*, *199, 200*, *209*, *214, 215*, *235, 236*, *240*, *249*, *255*, *257*, *261*
common_args, *136*
coord_flip, *207*
CorPairsPlot, 43
CorPairsPlotAtomic, *47*
CorPlot, 48
CorPlotAtomic, *48*, *52*

DensityPlot, *45*, 53
detect_venn_datatype, *247*
dim_example, 77
DimPlot, 61, *245*
DotPlot, 78
DotPlotAtomic, *85*
droplevels, *240*

element_grob.element_textbox
(element_textbox), 87
element_textbox, 87
element_textbox(), *89*
enrich_example, 95
enrich_multidb_example, 96
EnrichMap, 89, *91*
EnrichMapAtomic, *94*
EnrichNetwork (EnrichMap), 89
EnrichNetworkAtomic, *95*

facet_plot, *131*, *227*
FeatureDimPlot, *245*
FeatureDimPlot (DimPlot), 61
filter, *213*
fortify, *183*

geom_alluvium, *206*
geom_errorbar, *129*
geom_flow, *206*
geom_label, *207*
geom_point, *49*
geom_segment, *245*
geom_streamline, *71*, *243–245*
geom_text_repel, *124*, *155*, *238*, *247*, *250*
geom_text_wordcloud, *257*
ggplot2::alpha(), *163*
ggplot2::element_line(), *88*
ggplot2::margin(), *89*
ggplot2::theme(), *89*
ggraph, *150*
gridtext::textbox_grob(), *88, 89*
gsea_example, 101
GSEAPlot (GSEASummaryPlot), 96
GSEASummaryPlot, 96, *98*

Heatmap, 101, *109, 110*, *138–141*
HeatmapAnnotation, *108*
HeatmapAtomic, *101*, *110, 111*, *137–139*
Histogram, *45*
Histogram (DensityPlot), 53

igraph, *150*
iNEXT, *180*, *183*

JitterPlot, 119
JitterPlotAtomic, *125*

LinePlot, 126
LinePlotAtomic, *132*
LinkedHeatmap, *111*, 133
LinkedHeatmapAtomic, *140*
LollipopPlot (DotPlot), 78

ManhattanPlot, 142
ManhattanPlotAtomic, *147*
match.arg, *106*, *136*, *153, 154*
mean, *110*, *140*

Network, *56*, *91*, *135*, 150, *152*, *169*, *186*, *205*, *258*
NetworkAtomic, *156*

optimal.cutpoints, *196*

palette_list, 158, *217*
palette_this, *71*, 162, *206, 207*, *243, 244*
patchwork, *150*
patchwork::wrap_plots(), *58*, *73, 74*, *93, 94*, *110*, *140*, *155, 156*, *171*, *188, 189*, *208*, *260*
PieChart, 163, *190*, *194*
PieChartAtomic, *167*
position_dodge, *28*
position_dodge2, *14*
prepare_enrichr_result, *89*, *94*
prepare_upset_data, *237*
process_heatmap_data, *101*
process_keep_na_empty, *6*, *131*, *167*, *178*, *193*, *236*
process_theme, *166*, *172*, *215*, *261*

QQPlot, 168
QQPlotAtomic, *172*
quantile, *14*, *74*, *83*, *123*, *213*, *224*

RadarPlot, *173*, 173
RadarPlotAtomic, *173*, *179*
RarefactionPlot, 180
RarefactionPlotAtomic, *183*
RidgePlot, 184
RingPlot, 190
RingPlotAtomic, *193*
ROCCurve, 194
ROCCurveAtomic, *194*, *200*

SankeyPlot, 202, *205*
SankeyPlotAtomic, *209*
scale_fill_manual, *206*
scale_y_continuous, *123*

ScatterPlot, 210
ScatterPlotAtomic, *211*, *215*
shape, *153*
show_palettes, *106*, *108*, *136*, 216
SpatImagePlot, 217
SpatMasksPlot *(*SpatImagePlot*)*, 217
SpatPointsPlot *(*SpatImagePlot*)*, 217
SpatShapesPlot *(*SpatImagePlot*)*, 217
SpiderPlot, *173*
SpiderPlot *(*RadarPlot*)*, 173
SplitBarPlot *(*BarPlot*)*, 8
SplitBarPlotAtomic, *17*
stat_pp_band, *170*
stat_pp_line, *170*
stat_pp_point, *170*
stat_qq_band, *170*
stat_qq_line, *170*
stat_qq_point, *170*
StatStratum, *207*
symnum, *27*

theme, *230, 231*
theme_blank, 230
theme_box, 231
theme_this, 232
to_lodes_form, *202*
TrendPlot, 232
TrendPlotAtomic, *236*

unit, *154*
UpsetPlot, 237

validate_common_args, *6*, *51*, *94*, *99*, *124*, *125*, *131*, *146*, *166*, *172*, *178*, *182*, *183*, *193*, *199*, *200*, *214*, *215*, *235*, *239*, *260*
VelocityPlot, *75*, 241
VennDiagram, 246
VennDiagramAtomic, *249*
ViolinPlot, *45*
ViolinPlot *(*BoxPlot*)*, 20
VolcanoPlot, 250
VolcanoPlotAtomic, *255*

WaterfallPlot *(*BarPlot*)*, 8
WordCloudPlot, 257
WordCloudPlotAtomic, *261*
words_excluded, 262
wrap_plots, *6*, *51*, *111*, *124*, *130*, *131*, *141*, *166*, *178*, *182*, *193*, *194*, *199*, *235*, *239*, *240*, *248*, *255*

xlim, *252*