Package 'ggrecipes'

May 8, 2026

Title Recipes for Data Visualization

Version 0.1.0

Description A collection of custom 'ggplot2'-based visualizations for data exploration and analysis. Each function handles data preprocessing and returns a object that can be further customized using standard 'ggplot2' syntax.

License MIT + file LICENSE

URL https://github.com/Ignophi/ggrecipes, https://ignophi.github.io/ggrecipes/

BugReports https://github.com/Ignophi/ggrecipes/issues

Imports ggplot2, patchwork, ggrepel, scales, lifecycle

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Ignophi Hu [aut, cre, cph] (ORCID: <https://orcid.org/0009-0002-4094-0868>)

Maintainer Ignophi Hu <ignophi.hu@pm.me>

Repository CRAN

Date/Publication 2025-12-18 14:10:08 UTC

Contents

`gg_biodist` *Biodistribution Barplot with Optional Free-Scale Faceting*

Description

Creates a barplot visualization of biodistribution data (e.g., %ID/g across organs) with optional separation of specific organs onto free y-scales to prevent squishing of lower values. Points are overlaid on bars and all facets are displayed in a single row.

The function accepts:

- Long data: `value` is the name of the numeric column containing the measurements (original behaviour).
- Wide data: `value` is a regular expression pattern that matches the measurement columns (e.g. `"_val$"` for `Blood_val`, `Liver_val`, etc.). In this case, the data are internally converted to long format, creating a column named by `id` for organ/tissue and a column named by `value` for the measurements.

Usage

```
gg_biodist(
  data,
  id = "id",
  value = "value",
  y_label = value,
  group = NULL,
  separate = NULL,
  fill_colors = NULL,
  bar_alpha = 0.7,
  point_size = 1.5,
  stat_summary = "mean",
  error_bars = FALSE,
  quiet = FALSE
)
```

Arguments

`data`	A data frame containing biodistribution measurements.
`id`	Character string specifying the column name in `data` that contains organ/tissue identifiers (in long format) or the name that will be used for the organ/tissue column created from wide data. Default is "id".

value	Character string specifying either: (1) the column name in data that contains the measurement values (long data), or (2) a regular expression pattern that identifies the measurement columns (wide data, e.g., "_val$" to match Blood_val, Liver_val, etc.). Default is "value".
y_label	Character string specifying the label to use for the y-axis. Default is the same as value.
group	Character string specifying the column name in data that defines groups (e.g., treated vs control). Default is NULL (no grouping). When provided, bars and points are dodged and colored by group.
separate	Character vector of organ/tissue names (matching values in the id column) to display on separate y-axis scales. This prevents high-uptake organs from compressing visualization of lower-uptake organs. Default is NULL (no separation).
fill_colors	Character vector of colors used to fill bars and points. If group is NULL, the first color is used (default "#92b9de" if fill_colors is NULL). If group is provided, one color per group level is used (in the order of factor levels). If fill_colors is NULL in the grouped case, a default brewer palette ("Paired") is used.
bar_alpha	Numeric value (0-1) specifying transparency of bars. Default is 0.7.
point_size	Numeric value specifying size of points. Default is 1.5. If set to 0, points are not drawn.
stat_summary	Character string specifying summary statistic for bars. One of "mean" or "median". Default is "mean". Points show individual values.
error_bars	Logical indicating whether to add error bars (mean ± SD). Default is FALSE.
quiet	Logical indicating whether to suppress messages during data processing (e.g., when coercing non-numeric columns to numeric). Default is FALSE.

Details

The function automatically handles both long and wide format data:

- Long format: Each row represents one measurement with columns for organ ID and value
- Wide format: Each row represents one sample/replicate with separate columns for each organ (detected via regex pattern)

When separate is specified, high-uptake organs are displayed in separate facets with independent y-axis scales, preventing compression of lower values in the main plot.

Value

A ggplot2 object showing the biodistribution as barplot with points. The plot displays:

- Bars representing summary statistics (mean or median) for each organ
- Individual data points overlaid on bars
- Optional grouping with dodged bars and colored by group
- Optional separation of high-uptake organs onto independent y-scales
- Optional error bars showing standard deviation

Examples

```
bio_data <- data.frame(
  id        = paste0("sample_", 1:6),
  condition = rep(c("Control", "Treated"), each = 3),
  replicate = rep(1:3, times = 2),

  Blood_val  = c(4.8, 5.2, 4.5, 4.1, 4.3, 4.0),
  Heart_val  = c(1.9, 2.1, 2.0, 1.6, 1.8, 1.7),
  Lung_val   = c(3.5, 3.8, 3.2, 3.0, 3.1, 2.9),
  Liver_val  = c(14.2, 15.1, 13.8, 11.5, 12.0, 11.2),
  Spleen_val = c(9.1, 8.7, 9.4, 7.2, 7.5, 7.0),
  Kidney_val = c(125.0, 112.8, 121.9, 111.1, 102.4, 103.0),
  Tumor_val  = c(22.5, 24.1, 23.3, 28.2, 29.5, 27.8),
  Muscle_val = c(0.7, 0.6, 0.8, 0.5, 0.4, 0.6),
  Bone_val   = c(1.4, 1.6, 1.5, 1.1, 1.2, 1.0)
)

# Base biodist plot
gg_biodist(bio_data, id = "organ",
           value = "_val", group = "condition",
           point_size = 1.25,
           y_label = "%ID/g")

# Separate high uptake organs on separate axis
gg_biodist(bio_data, id = "organ",
           value = "_val", group = "condition",
           point_size = 1.25,
           y_label = "%ID/g",
           separate = c("Tumor", "Kidney"))

# Customization
gg_biodist(bio_data, id = "organ",
           value = "_val", group = "condition",
           point_size = 0, error_bars = TRUE,
           fill_colors = c("#e41a1c", "#377eb8"),
           y_label = "%ID/g",
           separate = c("Tumor", "Kidney"))
```

---

gg_conf *Confusion/Contingency Table Bubble Plot*

Description

The function automatically computes frequency counts for each unique combination of the x and y categorical variables using `table()`. Bubble sizes are scaled proportionally to represent counts, with the range controlled by `point_size_range`. Useful for visualizing cross-tabulations, confusion matrices, or any bivariate categorical data.

Usage

```
gg_conf(
  data,
  x,
  y,
  fill = "skyblue",
  text_size = 4,
  text_color = "black",
  point_size_range = c(3, 15),
  border_color = "black",
  show_grid = TRUE,
  expand = 0.15,
  facet_x = NULL,
  facet_y = NULL
)
```

Arguments

`data`	A data frame containing the categorical variables.
`x`	Character string specifying the column name in `data` for the x-axis categorical variable.
`y`	Character string specifying the column name in `data` for the y-axis categorical variable.
`fill`	Character string specifying the fill color for bubbles. Default is "skyblue".
`text_size`	Numeric value specifying the size of count labels. Default is 4.
`text_color`	Character string specifying the color of count labels. Default is "black".
`point_size_range`	Numeric vector of length 2 specifying the minimum and maximum bubble sizes. Default is `c(3, 15)`.
`border_color`	Character string specifying the color of bubble borders. Default is "black".
`show_grid`	Logical indicating whether to show major grid lines. Default is TRUE.
`expand`	Numeric value specifying the expansion multiplier for both axes. Default is 0.15.
`facet_x`	Character string specifying an optional column name in `data` for horizontal faceting. Default is NULL (no faceting).
`facet_y`	Character string specifying an optional column name in `data` for vertical faceting. Default is NULL (no faceting).

Value

A ggplot2 object showing the confusion table as a bubble plot. The plot displays:

- Bubbles at each x-y combination, sized by frequency count
- Count labels displayed in the center of each bubble
- Optional faceting for additional categorical variables
- Customizable colors, sizes, and grid visibility

Examples

```
data(mtcars)
# Create synthetic categorical variables
# Bin horsepower & fuel efficiency into ordered categories
mtcars$horsepower <-
  cut(mtcars$hp, breaks = 5,
      labels = c("Very Low", "Low", "Medium", "High", "Very High"))
mtcars$`miles per gallon` <-
  cut(mtcars$mpg, breaks = 5,
      labels = c("Very Low", "Low", "Medium", "High", "Very High"))

# Base plot
gg_conf(data = mtcars, x = "horsepower", y = "miles per gallon")

# Custom styling
gg_conf(data = mtcars, x = "horsepower", y = "miles per gallon",
        fill = "lightcoral", point_size_range = c(5, 20),
        show_grid = FALSE)

# With faceting by "vs" column
gg_conf(data = mtcars, x = "horsepower", y = "miles per gallon",
        fill = "lightcoral", point_size_range = c(5, 20),
        facet_x = "vs",
        show_grid = FALSE)
```

---

gg_criteria *Criteria Heatmap with Optional Barplots*

Description

Creates a tile-based heatmap visualization where samples are displayed on the y-axis and criteria are shown on the x-axis. Each tile shows a criterion value with color coding. Optional barplots can be added to the right side to display numeric metrics for each sample.

Usage

```
gg_criteria(
  data,
  id = "id",
  criteria = "_criteria",
  bar_column = NULL,
  bar_fill = NULL,
  panel_ratio = 0.3,
  tile_width = 0.7,
  tile_height = 0.7,
  tile_alpha = 1,
  tile_fill = NULL,
  show_text = TRUE,
```

```
  border_color = "black",
  border_width = 0.5,
  text_size = 8,
  show_legend = TRUE,
  quiet = FALSE
)
```

Arguments

data	A data frame containing the data to visualize.
id	Character string specifying the column name in data that contains sample identifiers. Default is "id".
criteria	Character string specifying either: (1) the column name in data that contains criterion names (long data, not yet supported), or (2) a regular expression pattern that identifies the criterion columns (wide data, e.g., "_criteria$" to match Pass_criteria, Fail_criteria, etc.). Default is "_criteria".
bar_column	Character vector specifying column name(s) in data to display as horizontal barplot(s) to the right of the heatmap. Columns must be numeric. Default is NULL (no barplots).
bar_fill	Character vector specifying the fill color(s) for bars. When bar_column contains multiple elements, colors are recycled if necessary to match the number of bars. If NULL (default), uses colors from the Brewer "Paired" palette.
panel_ratio	Numeric value specifying the total relative width of the barplot panel(s) compared to the heatmap panel (which has a reference width of 1). When multiple bar_column values are provided, this width is divided equally among them. For example, panel_ratio = 0.3 creates a 1:0.3 width ratio between heatmap and barplot panels. Only used when bar_column is not NULL. Default is 0.3.
tile_width	Numeric value (0-1) specifying the width of tiles as a proportion of available space. Default is 0.7.
tile_height	Numeric value (0-1) specifying the height of tiles as a proportion of available space. Default is 0.7.
tile_alpha	Numeric value (0-1) specifying transparency of tiles. Default is 1 (fully opaque).
tile_fill	Named character vector of colors for criterion values (e.g., c(Pass = "green", Fail = "red")). If NULL (default), uses the Brewer "Paired" palette.
show_text	Logical indicating whether to show criterion values as text labels on tiles. Default is TRUE.
border_color	Character string specifying the color of tile borders. Default is "black".
border_width	Numeric value specifying the width of tile borders. Default is 0.5.
text_size	Numeric value specifying the size of axis text. Default is 8.
show_legend	Logical indicating whether to show the fill legend. Default is TRUE.
quiet	Logical indicating whether to suppress messages. Default is FALSE.

Details

The function currently supports wide format data where each criterion is a separate column. The column names should match the pattern specified in `criteria` (e.g., columns ending in "_crit"). The pattern is removed from column names to create criterion labels.

When `bar_column` is specified, horizontal barplots are added to the right side of the heatmap. Multiple barplots can be created by providing a vector of column names. Each barplot shows the numeric values with text labels positioned outside the bars.

The function calculates and suggests plot dimensions based on the number of samples and criteria. Access these via `attr(plot, "recommended_dims")`.

WARNING: Alignment between heatmap and barplots depends on plot dimensions. The function provides recommended dimensions (accessible via `attr(plot, "recommended_dims")`) that ensure proper alignment. You can adjust these dimensions to improve appearance (e.g., reduce width to tighten spacing, or scale proportionally for size) while maintaining alignment.

Value

A ggplot2 object (or patchwork object if `bar_column` is used) showing the criteria heatmap. The plot displays:

- Tiles at each sample-criterion intersection
- Tile colors indicating criterion values
- Criterion values as text labels within tiles
- Optional horizontal barplots on the right side
- A "recommended_dims" attribute with suggested width and height in inches

Examples

```
# Example: Gene prioritization criteria
gene_data <- data.frame(
  gene = c("BRCA1", "TP53", "EGFR", "KRAS", "MYC",
           "PTEN", "APC", "CDKN2A", "RB1", "VHL"),
  `Missense Variant_crit` = c("Yes", "Yes", "Yes", NA, "Yes",
                              "Yes", NA, "Yes", NA, "Yes"),
  `eQTL_crit` = c("Yes", "Yes", NA, "Yes", "Yes",
                  "Yes", "Yes", "Yes", "Yes", NA),
  `pQTL_crit` = c("Yes", NA, "Yes", "Yes", NA,
                  "Yes", "Yes", NA, "Yes", "Yes"),
  `GWAS Hit_crit` = c("Yes", "Yes", "Yes", "Yes", NA,
                      "Yes", "Yes", "Yes", NA, NA),
  `Loss of Function_crit` = c(NA, "Yes", NA, NA, "Yes",
                              "Yes", "Yes", NA, "Yes", NA),
  `High Conservation_crit` = c("Yes", "Yes", "Yes", "Yes", "Yes",
                               "Yes", "Yes", "Yes", "Yes", "Yes"),
  `mRNA DE_crit` = c("Yes", NA, "Yes", NA, NA,
                     "Yes", "Yes", "Yes", NA, "Yes"),
  `Prot DE_crit` = c(NA, "Yes", NA, NA, NA,
                     "Yes", NA, NA, NA, "Yes"),
  check.names = FALSE
```

```
)

# Calculate total criteria met
crit_cols <- grep("_crit$", names(gene_data), value = TRUE)
gene_data$`Total` <- rowSums(gene_data[crit_cols] == "Yes", na.rm = TRUE)

# Base criteria plot
# **Warn**: space between heatmap and barplots depends on plot width
gg_criteria(
  data = gene_data,
  id = "gene",
  criteria = "_crit$",
  bar_column = "Total",
  show_text = FALSE,
  tile_fill = c(Yes = "#A6CEE3", No = "white"),
  bar_fill = "#A6CEE3",
  panel_ratio = 1
)

# Example: VHH Variant Analysis
# Define amino acid chemistry colors
aa_colors <- c(
  "D" = "#E60A0A", "E" = "#E60A0A", # Acidic (red)
  "K" = "#145AFF", "R" = "#145AFF", # Basic (blue)
  "H" = "#8282D2",                  # Histidine (purple)
  "S" = "#FA9600", "T" = "#FA9600", # Polar uncharged (orange)
  "N" = "#00DCDC", "Q" = "#00DCDC", # Polar amides (cyan)
  "C" = "#E6E600",                  # Cysteine (yellow)
  "G" = "#EBEBEB",                  # Glycine (light gray)
  "P" = "#DC9682",                  # Proline (tan)
  "A" = "#C8C8C8",                  # Alanine (gray)
  "V" = "#0F820F", "I" = "#0F820F", # Hydrophobic (green)
  "L" = "#0F820F", "M" = "#0F820F",
  "F" = "#3232AA", "W" = "#B45AB4", # Aromatic (dark blue/purple)
  "Y" = "#3232AA"
)

vhh_variants <- data.frame(
  variant = c("WT", "Mut1", "Mut2", "Mut3", "Mut4", "Mut7", "Mut5",
              "Mut6", "Mut8", "Mut9", "Mut10", "Mut11"),
  Q5_mut = c(NA, "H", NA, NA, NA, NA, "H", "H", "H", "D", NA, "H"),
  S55_mut = c(NA, NA, "P", NA, NA, "P", "P", NA, "P", "P", NA, NA),
  N73_mut = c(NA, NA, NA, "E", NA, NA, NA, "E", "E", NA, "E", NA),
  K80_mut = c(NA, "L", NA, NA, "S", "V", NA, NA, NA, "L", "S", NA),
  F99_mut = c(NA, NA, "L", NA, NA, NA, NA, NA, NA, "W", NA, "W"),
  KD_nM = c(45, 18, 5.2, 38, 42, 20, 3.8, 15, 3.2, 4.5, 40, 22),
  yield_mg_L = c(12, 11.8, 10, 13, 11, 10, 10, 12, 10, 7.8, 12.5, 8.5),
  Tm_C = c(68.5, 67.8, 68, 72.3, 35, 66, 67.5, 70, 70.5, 72, 38, 74)
)

# Create the systematic variant heatmap
gg_criteria(
  data = vhh_variants,
```

```
  id = "variant",
  criteria = "_mut$",
  tile_fill = aa_colors,
  bar_column = c("KD_nM", "yield_mg_L", "Tm_C"),
  panel_ratio = 2,
  tile_width = 0.70,
  tile_height = 0.70,
  show_text = TRUE,
  border_color = "grey40",
  border_width = 0.4,
  text_size = 10,
  show_legend = FALSE
)
```

---

gg_geno *Genotype Heatmap with Optional Barplots*

Description

Creates a heatmap visualization of biallelic genotypes (e.g., SNPs, variants) with split-tile representation showing phased or unphased alleles. Each tile is divided diagonally to display both alleles, with border color indicating phasing status (black for phased, white for unphased). Optional barplots can be added to display associated numeric data.

Usage

```
gg_geno(
  data,
  id = "id",
  geno = "_geno",
  bar_column = NULL,
  bar_fill = NULL,
  panel_ratio = 0.3,
  tile_fill = NULL,
  tile_width = 0.7,
  tile_height = 0.7,
  border_width = 0.5,
  text_size = 10,
  show_legend = TRUE,
  quiet = FALSE
)
```

Arguments

data	A data frame containing the data to visualize.
id	Character string specifying the column name in data that contains sample identifiers. Default is "id".

geno	Character string specifying the regular expression pattern that identifies genotype columns (e.g., "_geno$" to match columns like rs123_geno, rs456_geno). The pattern is removed to create SNP labels. Default is "_geno".
bar_column	Character vector specifying column name(s) in data to display as horizontal barplot(s) to the right of the heatmap. Columns must be numeric. Default is NULL (no barplots).
bar_fill	Character vector specifying the fill color(s) for bars. When bar_column contains multiple elements, colors are recycled if necessary to match the number of bars. If NULL (default), uses colors from the Brewer "Paired" palette.
panel_ratio	Numeric value specifying the total relative width of the barplot panel(s) compared to the heatmap panel (which has a reference width of 1). When multiple bar_column values are provided, this width is divided equally among them. For example, panel_ratio = 0.3 creates a 1:0.3 width ratio between heatmap and barplot panels. Only used when bar_column is not NULL. Default is 0.3.
tile_fill	Named character vector of colors for criterion values (e.g., c(Pass = "green", Fail = "red")). If NULL (default), uses the Brewer "Paired" palette.
tile_width	Numeric value (0-1) specifying the width of tiles as a proportion of available space. Default is 0.7.
tile_height	Numeric value (0-1) specifying the height of tiles as a proportion of available space. Default is 0.7.
border_width	Numeric value specifying the width of tile borders. Default is 0.5.
text_size	Numeric value specifying the size of axis text. Default is 8.
show_legend	Logical indicating whether to show the fill legend. Default is TRUE.
quiet	Logical indicating whether to suppress messages. Default is FALSE.

Details

The function expects genotype data in wide format with:

- One column for sample IDs (specified by id)
- Multiple columns matching the geno pattern, each representing a SNP/variant
- Genotypes encoded as "allele1/allele2" (unphased) or "allele1|allele2" (phased)

Genotype format examples:

- Unphased: "0/0", "0/1", "1/1"
- Phased: "0|0", "0|1", "1|0"

Each genotype tile is split diagonally with the first allele in the top-left triangle and the second allele in the bottom-right triangle. The border color indicates phasing: black borders for phased genotypes (|) and white borders for unphased genotypes (/).

When bar_column is specified, the function requires the patchwork package to combine the heatmap with barplots. Barplots are added to the right of the heatmap and display numeric values with text labels.

WARNING: Alignment between heatmap and barplots depends on plot dimensions. The function provides recommended dimensions (accessible via attr(plot, "recommended_dims")) that ensure proper alignment. You can adjust these dimensions to improve appearance (e.g., reduce width to tighten spacing, or scale proportionally for size) while maintaining alignment.

Value

A ggplot2 object (or patchwork object if `bar_column` is specified). The plot displays:

- Split tiles representing biallelic genotypes
- Top-left triangle: first allele
- Bottom-right triangle: second allele
- Black borders: phased genotypes (separator: |)
- White borders: unphased genotypes (separator: /)
- Optional barplots showing numeric data for each sample
- Color legend mapping alleles to colors

The returned object has an attribute `"recommended_dims"` containing suggested plot width and height in inches.

Examples

```
# Create example SNP and phenotype data
set.seed(123)

snp_data <- data.frame(
  id = paste0("P", sprintf("%03d", 1:12)),
  # SNP columns
  rs1234_geno = sample(c(c("0/0", "0/1", "1/1"), NA),
                       12, replace = TRUE,
                       prob = c(0.4, 0.4, 0.15, 0.05)),
  rs5678_geno = sample(c("0/0", "0/1", "0/2", "1/1", "1/2", "2/2", NA),
                       12, replace = TRUE,
                       prob = c(0.25, 0.25, 0.1, 0.15, 0.15, 0.05, 0.05)),
  rs9012_geno = sample(c(c("0|0", "0|1", "1|1", "0/1", "1/2"), NA),
                       12, replace = TRUE,
                       prob = c(0.2, 0.2, 0.15, 0.2, 0.15, 0.1)),
  rs3456_geno = sample(c(c("0/0", "0/1", "1/1"), NA),
                       12, replace = TRUE,
                       prob = c(0.45, 0.35, 0.15, 0.05)),
  rs7890_geno = sample(c("0/0", "0/1", "0/2", "1/3", "2/2", NA),
                       12, replace = TRUE,
                       prob = c(0.3, 0.25, 0.15, 0.1, 0.15, 0.05)),
  rs2468_geno = sample(c("0|0", "0|1", "1|1", "1|2", NA),
                       12, replace = TRUE,
                       prob = c(0.3, 0.35, 0.2, 0.1, 0.05)),

  # Phenotype columns for bar plots
  Age = sample(25:75, 12, replace = TRUE),
  BMI = round(rnorm(12, mean = 26, sd = 4), 1),
  Insulin = round(rnorm(12, mean = 12, sd = 3), 1)
)

# Base genotype plot
gg_geno(
  data = snp_data,
```

```
  id = "id",
  geno = "_geno$"
)

# Show optional barplots
gg_geno(
  data = snp_data,
  id = "id",
  geno = "_geno$",
  show_legend = TRUE,
  panel_ratio = 1,
  bar_column = c("Age", "BMI", "Insulin"),
  bar_fill = c("#c77d77", "#e0b46e", "#c7bc77"),
  text_size = 10
)
```

---

gg_kdmap *Kinetic Rate Map (Association/Dissociation plot)*

Description

Generates a log-log plot of association rate (ka) vs dissociation rate (kd) with iso-affinity (KD) lines. Useful for visualizing kinetic binding data from surface plasmon resonance (SPR), biolayer interferometry (BLI), or other biophysical assays.

Usage

```
gg_kdmap(
  data,
  id = "id",
  ka = "ka",
  kd = "kd",
  labels = NULL,
  size = 4,
  shape = 21,
  fill = "grey",
  color = "black",
  ref_id = NULL,
  ref_shape = 21,
  ref_color = "black",
  ref_fill = "white",
  ref_size = size,
  rep_lines = TRUE,
  iso_color = "black",
  iso_alpha = 1,
  iso_width = 0.3,
  iso_type = "dashed",
  iso_n = 8,
```

```
  text_padding = 1,
  title = NULL,
  show_anno = FALSE
)
```

Arguments

data	A data frame containing kinetic data. Must include columns specified by the id, ka, and kd parameters.
id	Character string specifying the column name in data that contains identifiers for each measurement. Used to group replicates and identify reference points. Default is "id".
ka	Character string specifying the column name in data that contains association rate constants. Values must be in M^-1s^-1 units. Default is "ka".
kd	Character string specifying the column name in data that contains dissociation rate constants. Values must be in s^-1 units. Default is "kd".
labels	Character string specifying a column name in data containing text labels to display next to points. If NULL (default), no labels are shown.
size	Numeric value for point size, or character string specifying a column name in data to map to point size. Default is 3.
shape	Numeric value (0-25) for point shape, or character string specifying a column name in data to map to point shape. Default is 21 (filled circle).
fill	Character string specifying a color for point fill, or a column name in data to map to fill aesthetic. Default is "grey".
color	Character string specifying a color for point border, or a column name in data to map to color aesthetic. Default is "black".
ref_id	Character string specifying the ID of a reference point to highlight with different aesthetics. Must match a value in the column specified by id. Default is NULL (no reference point).
ref_shape	Numeric value (0-25) specifying the shape for the reference point. Default is 21.
ref_color	Character string specifying the border color for the reference point. Default is "black".
ref_fill	Character string specifying the fill color for the reference point. Default is "white".
ref_size	Numeric value specifying the size for the reference point. Default is the value of the size parameter.
rep_lines	Logical indicating whether to connect replicate points (points with the same ID) with lines. Default is TRUE.
iso_color	Character string specifying the color of iso-KD lines. Default is "black".
iso_alpha	Numeric value (0-1) specifying the transparency of iso-KD lines. Default is 1.
iso_width	Numeric value specifying the line width of iso-KD lines. Default is 0.3.
iso_type	Character string specifying the line type of iso-KD lines. Must be one of "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash". Default is "dashed".

iso_n	Numeric value specifying the number of iso-KD lines to draw, spaced evenly in log10(KD) across the current plot range. Default is 8.
text_padding	Numeric value specifying the padding around text labels when labels is used. Passed to ggrepel::geom_text_repel. Default is 1.
title	Character string for the plot title. If NULL (default), no title is displayed.
show_anno	Logical indicating whether to show corner annotations indicating fast on-rate (top left) and fast off-rate (bottom right) regions. Default is FALSE.

Details

The function creates a log-log plot with:

- X-axis: dissociation rate (kd, in s^-1)
- Y-axis: association rate (ka, in M^-1s^-1)
- Diagonal lines representing iso-affinity contours (constant KD values)

Required units:

- ka: M^-1s^-1 (association rate constant)
- kd: s^-1 (dissociation rate constant)

The function calculates equilibrium dissociation constant as KD = kd/ka and automatically converts to the most appropriate units (pM, nM, μM, or mM) for display on secondary axes.

Iso-KD lines are generated so that there are always iso_n lines spanning the visible plot window. They are equally spaced in log10(KD) across the KD range implied by the current x/y limits. The top (x) and right (y) secondary axes are labeled with the KD values corresponding to these lines.

Value

A ggplot2 object showing the kinetic rate map. The plot displays:

- X-axis: dissociation rate (kd) in s^-1 on log scale
- Y-axis: association rate (ka) in M^-1s^-1 on log scale
- Points representing individual kinetic measurements
- Diagonal iso-affinity lines representing constant KD values
- Secondary axes (top and right) showing KD values in appropriate units (pM, nM, μM, or mM)
- Optional connecting lines between replicates (same ID)
- Optional highlighted reference point
- Optional text labels for points
- Optional corner annotations indicating rate directions

KD values are automatically calculated from ka and kd rates and displayed in the most appropriate units based on the data range.

See Also

geom_point for point customization, geom_text_repel for label placement.

Examples

```
# Basic example: 5 variants with single measurements
kinetic_data <- data.frame(
  id = c("WT", "Mut1", "Mut2", "Mut3", "Mut4"),
  ka = c(1.2e5, 2.5e5, 2e5, 8.0e4, 1.8e5),
  kd = c(1.5e-3, 2.0e-3, 1.5e-3, 1.2e-3, 1.8e-3)
)

gg_kdmap(data = kinetic_data, show_anno = TRUE)

# With replicates: lines connect points with same ID
kinetic_rep <- data.frame(
  id = c("WT", "WT", "WT", "Mut1", "Mut1", "Mut2", "Mut3", "Mut4"),
  ka = c(1.2e5, 1.5e5, 1.1e5, 2.5e5, 2.4e5, 2e5, 8.0e4, 1.8e5),
  kd = c(1.5e-3, 1.6e-3, 1.4e-3, 2.0e-3, 1.9e-3, 1.5e-3, 1.2e-3, 1.8e-3)
)

gg_kdmap(data = kinetic_rep, show_anno = TRUE, fill = "id")

# Add labels and highlight reference
gg_kdmap(
  data = kinetic_rep,
  labels = "id",
  ref_id = "WT",
  ref_fill = "white",
  ref_color = "red",
  fill = "id"
)

# Customize iso-KD lines
gg_kdmap(
  data = kinetic_rep,
  iso_n = 12,
  iso_color = "#7192ad",
  iso_type = "solid"
)

# Turn off replicate lines
gg_kdmap(data = kinetic_rep, rep_lines = FALSE)

# Custom column names
custom_data <- data.frame(
  sample = c("WT", "Mut1", "Mut2"),
  kon = c(1.2e5, 2.5e5, 5.0e4),
  koff = c(1.5e-3, 2.0e-3, 5.0e-4)
)

gg_kdmap(data = custom_data, id = "sample", ka = "kon", kd = "koff")
```

---

gg_rankshift *Rank Shift Plot*

Description

Creates a three-panel visualization comparing ranks of samples between two groups. Left and right panels show distributions (bars or boxplots) for each group ordered by rank. The center panel displays connecting lines colored by rank change direction, revealing which samples improved, declined, or maintained their relative position.

Usage

```
gg_rankshift(
  data,
  id = "id",
  group = "group",
  value = "value",
  style = "box",
  fill = NULL,
  alpha = 0.7,
  line_alpha = 0.5,
  line_width = 0.5,
  panel_ratio = 0.5,
  text_size = 11,
  show_points = TRUE,
  point_size = 1,
  point_shape = 21,
  point_alpha = 0.7,
  stat_summary = "mean",
  decreasing = FALSE,
  free_x = TRUE,
  rank_change_colors = c(increase = "#d73027", decrease = "#4575b4", no_change =
    "grey50")
)
```

Arguments

`data`	A data frame containing the data to visualize.
`id`	Character string specifying the column name in `data` that contains sample identifiers. Default is "id".
`group`	Character string specifying the column name in `data` that defines the two groups to compare. Must contain exactly 2 unique values. Default is "group".
`value`	Character string specifying the column name in `data` that contains the numeric values used for ranking. Default is "value".
`style`	Character string specifying visualization type: `"bar"` for barplots or `"box"` for boxplots. Default is "box".
`fill`	Character vector of length 2 specifying fill colors for the two groups. If NULL (default), uses `c("#92b9de", "#e8927c")`.
`alpha`	Numeric value (0-1) specifying transparency of bars/boxes. Default is 0.7.
`line_alpha`	Numeric value (0-1) specifying transparency of connecting lines. Default is 0.5.

`line_width` Numeric value specifying width of connecting lines. Default is 0.3.

`panel_ratio` Numeric value specifying the relative width of the center line panel compared to side panels (which have width 1). Default is 1.

`text_size` Numeric value specifying the base size for axis text. Default is 11.

`show_points` Logical indicating whether to overlay individual data points on bars/boxes. Default is TRUE.

`point_size` Numeric value specifying size of points when `show_points = TRUE`. Default is 1.

`point_shape` Numeric value (0-25) specifying point shape when `show_points = TRUE`. Default is 21 (filled circle).

`point_alpha` Numeric value (0-1) specifying transparency of points when `show_points = TRUE`. Default is 0.7.

`stat_summary` Character string specifying the summary statistic used for ranking: `"mean"` or `"median"`. Default is "mean".

`decreasing` Logical indicating rank order direction. If TRUE, higher values receive lower rank numbers (rank 1 = highest value). If FALSE, lower values receive lower rank numbers. Default is FALSE.

`free_x` Logical indicating whether x-axes are independent between panels. If FALSE, both panels share the same x-axis limits. Default is TRUE.

`rank_change_colors` Named character vector of length 3 specifying colors for rank changes. Must contain names `"increase"`, `"decrease"`, and `"no_change"`. Default is `c(increase = "#d73027", decrease = "#4575b4", no_change = "grey50")`.

Details

The function identifies samples common to both groups, calculates summary statistics (mean or median) for each sample within each group, and assigns ranks based on these values. Samples are then displayed in rank order with connecting lines showing how ranks changed between groups.

When `style = "box"` and `show_points = TRUE`, boxplots display the distribution while individual points show all replicates. This is useful for visualizing measurement variability alongside rank changes.

The `decreasing` parameter controls ranking direction:

- `FALSE` (default): Rank 1 = lowest value, higher ranks = higher values
- `TRUE`: Rank 1 = highest value, lower ranks = lower values

Value

A patchwork object combining three plots: left panel showing the first group's distribution, center panel showing rank change lines, and right panel showing the second group's distribution. Line colors indicate whether samples increased in rank (moved up), decreased in rank (moved down), or maintained the same rank between groups.

Examples

```
# Example data: bacterial strain growth rate control vs antibiotic treated
growth_data <- data.frame(
  strain = rep(paste0("Strain", 1:13), each = 6),
  condition = rep(c("Control", "Treated"), each = 3, times = 13),
  growth_rate = c(
    rnorm(39, mean = 0.85, sd = 0.12),  # Control
    rnorm(39, mean = 0.45, sd = 0.10)   # Treated (reduced growth)
  )
)

# Basic rank shift plot
gg_rankshift(
  data = growth_data,
  id = "strain",
  group = "condition",
  value = "growth_rate",
)

# With barplots instead of boxplots
gg_rankshift(
  data = growth_data,
  id = "strain",
  group = "condition",
  value = "growth_rate",
  style = "bar"
)

# Custom styling
gg_rankshift(
  data = growth_data,
  id = "strain",
  group = "condition",
  value = "growth_rate",
  fill = c("#e41a1c", "#377eb8"),
  rank_change_colors = c(
    increase = "#1b9e77",
    decrease = "#d95f02",
    no_change = "#7570b3"
  ),
  panel_ratio = 0.5,
  point_size = 2.5,
  line_width = 1,
  decreasing = TRUE
)
```

---

gg_seq *Sequence Coverage Plot*

Description

Plots sequences that are substrings of a reference sequence, with each unique sequence shown as a row at its aligned position. Useful for visualizing peptide mapping coverage, proteomics experiments, or any analysis where you need to show which parts of a reference sequence are covered by shorter sequences. Supports character coloring and region highlighting (e.g., CDRs, tags, binding sites).

Usage

```
gg_seq(
  data,
  ref,
  sequence = "sequence",
  name = NULL,
  color = NULL,
  highlight = NULL,
  wrap = NULL,
  size = 3,
  face = "plain",
  show_ref = TRUE,
  margin_t = 30,
  annotate = NULL,
 annotate_defaults = list(angle = 0, vjust = 0.5, size = 3, face = "plain", color =
    "black")
)
```

Arguments

data	A data frame containing sequences.
ref	Character string of the reference sequence against which sequences will be aligned.
sequence	Character string specifying the column name in data that contains the sequences. Default is "sequence".
name	Character string specifying the column name in data that contains names/IDs for the sequences. If NULL (default), no names are shown on the y-axis. If specified, these names will be displayed instead of blank labels. Note: If the same sequence appears multiple times in data with different names, only the first occurrence's name will be used in the plot.
color	Named character vector of colors for specific characters, or NULL (default). If NULL, all characters are displayed in black. Characters specified here are automatically displayed in bold. Example: c(K = "blue", R = "red").
highlight	Named list where names are valid R colors and values are numeric vectors of positions to highlight with vertical shading. The shading spans the full height of the plot with semi-transparent color (alpha = 0.3) and black borders. Consecutive positions are automatically merged into continuous bands. To specify multiple regions with the same color, provide them as a single vector: list("#FFE0B2" = c(1:10, 50:60)). Default is NULL (no highlighting).

`wrap` Numeric value specifying the maximum number of positions to display per row before wrapping to the next panel. If NULL (default), no wrapping is applied and the entire sequence is displayed in a single row. When specified, the plot is split into multiple vertically stacked panels, each showing up to `wrap` positions. Useful for displaying long sequences in a more compact format.

`size` Numeric value for the size of sequence characters. Default is 3.

`face` Character string specifying font face for characters. Must be one of "plain", "bold", "italic", or "bold.italic". Default is "plain". Note: Characters specified in `color` are always displayed in bold regardless of this setting.

`show_ref` Logical indicating whether to show the reference sequence in the plot. Default is TRUE.

`margin_t` Numeric value for the top margin in points. Increases space above the plot to prevent annotation clipping. Adjust this value upward if rotated annotations are cut off. Default is 30.

`annotate` List of annotation specifications to add text labels above the plot. Each element must be a list with required elements `label` (character string) and `pos` (numeric position). Optional elements: `angle` (rotation angle), `hjust` (horizontal justification, defaults to 0 for 90-degree angles and 0.5 for other angles), `vjust` (vertical justification), `size` (text size), `face` (font face), `color` (text color). Default is NULL (no annotations).

`annotate_defaults` List of default values for annotation styling. Valid elements: `angle` (default 0), `vjust` (default 0.5), `size` (default 3), `face` (default "plain"), `color` (default "black"). These defaults are used when individual annotations don't specify these parameters. Default is `list(angle = 0, vjust = 0.5, size = 3, face = "plain", color = "black")`.

Value

A ggplot2 object showing the aligned sequences. The plot displays:

- Character sequences aligned horizontally by their position in the reference
- Each sequence on a separate row, ordered by starting position
- Optional reference sequence at the top (if `show_ref = TRUE`)
- Specified characters highlighted in bold with custom colors
- Optional vertical shading bands at specified positions with black borders
- Optional text annotations above the plot

If no sequences from `data` match the reference sequence, returns an empty ggplot2 object with a warning message. Sequences that don't match the reference are silently excluded from the plot without warning.

See Also

`gg_seqdiff` for visualizing sequence differences relative to a reference.

Examples

```
# Create synthetic example of peptide mapping data
# Reference sequence
ref_seq <- paste0(
  "QVQLVESGGGLVQAGGSLRLSCAASGFTFSSYAMGWFRQAPGKEREFVAAINSGGST",
  "YYPDSVKGRFTISRDNAKNTVYLQMNSLKPEDTAVYYCAADLRGTTVNNYWGQGTQV",
  "TVSSEQKLISEEDL"
)

# Peptides with RT and intensity
df_peptides <- data.frame(
  id = c("Pep_1004", "Pep_1010", "Pep_1007", "Pep_1011",
         "Pep_1009", "Pep_1005", "Pep_1013", "Pep_1003",
         "Pep_1001", "Pep_1012", "Pep_1006", "Pep_1008",
         "Pep_1002"),
  sequence = c(
    "QAPGKER",
    "GRFTISR",
    "GTTVNNYWGQGTQVTVSSEQKLISEEDL",
    "GRFTISRDNAKNTVYLQMNSLK",
    "EREFVAAINSGGSTYYPDSVK",
    "QAPGKEREFVAAINSGGSTYYPDSVKGR",
    "NTVYLQMNSLKPEDTAVYYCAADLR",
    "LSCAASGFTFSSYAMGWFRQAPGKER",
    "QVQLVESGGGLVQAGGSLR",
    "PEDTAVYYCAADLRGTTVNNYWGQGTQVTVSSEQKLISEEDL",
    "FTISRDNAKNTVYLQMNSLKPEDTAVYYCAADLR",
    "LSCAASGFTFSSYAMGWFRQAPGK",
    "LSCAASGFTFSSYAMGWFR"
  ),
  rt_min = c(10, 28.5, 34.4, 34.4, 36, 36.5, 40.8,
             42.5, 42.8, 43.3, 44.1, 44.8, 46.7),
  intensity = c(2769840, 2248170, 2172370, 1698280, 2202810,
                983267, 659246, 1064906, 1988932, 1438544,
                639990, 1017811, 1112824),
  stringsAsFactors = FALSE
)

# Base coverage map
gg_seq(data = df_peptides, ref = ref_seq, wrap = 70)

# With peptide IDs and residue coloring
gg_seq(
  data = df_peptides,
  ref = ref_seq,
  name = "id",
  color = c(C = "red", K = "blue", R = "#468c2d"),
  highlight = list(
    "#ffb4b4" = c(27:33, 51:57, 96:107),
    "#70bcfa" = c(1, 43, 64, 75, 86)
  ),
  wrap = 70
```

```
)

# With annotations
gg_seq(
  data = df_peptides,
  ref = ref_seq,
  name = "id",
  color = c(C = "red", K = "blue", R = "#468c2d"),
  highlight = list(
    "#ffb4b4" = c(27:33, 51:57, 96:107),  # CDR regions
    "#70bcfa" = c(1, 43, 64, 75, 86),     # Lysines
    "#d68718" = c(105:106),               # Liability site
    "#94d104" = c(119:128)                # c-Myc tag
  ),
  annotate = list(
    list(label = "CDR1", pos = 30),
    list(label = "CDR2", pos = 54),
    list(label = "CDR3", pos = 101),
    list(label = "N-term", pos = 1, angle = 90, vjust = 1),
    list(label = "K43", pos = 43, angle = 90),
    list(label = "K64", pos = 64, angle = 90),
    list(label = "K75", pos = 75, angle = 90),
    list(label = "K86", pos = 86, angle = 90),
    list(label = "Liability", pos = 106, angle = 90),
    list(label = "c-Myc tag", pos = 124)
  ),
  annotate_defaults = list(face = "bold"),
  wrap = 80
)
```

`gg_seqdiff` *Sequence Difference Plot*

Description

Visualizes mutations across aligned sequences by displaying only positions that differ from a reference sequence. Accepts either a data frame with sequences (which are aligned internally using Needleman-Wunsch global alignment) or a pre-aligned Clustal file. Supports character coloring and region highlighting (e.g., CDRs, tags, binding sites). Useful for focusing on differences in sequence comparisons. Inspired by Jalview's behavior when setting a reference sequence and hiding matching positions.

Usage

```
gg_seqdiff(
  data,
  ref,
  sequence = "sequence",
  name = NULL,
```

```
  clustal = NULL,
  color = NULL,
  highlight = NULL,
  wrap = NULL,
  size = 3,
  show_ref = TRUE,
  margin_t = 30,
  annotate = NULL,
 annotate_defaults = list(angle = 0, vjust = 0.5, size = 3, face = "plain", color =
    "black")
)
```

Arguments

data	A data frame containing sequences.
ref	Character string of the reference sequence against which sequences will be aligned.
sequence	Character string specifying the column name in data that contains the sequences. Default is "sequence".
name	Character string specifying the column name in data that contains names/IDs for the sequences. If NULL (default), no names are shown on the y-axis. If specified, these names will be displayed instead of blank labels. Note: If the same sequence appears multiple times in data with different names, only the first occurrence's name will be used in the plot.
clustal	Character string specifying path to a Clustal alignment file. If provided, overrides data, ref, sequence, and name. Default is NULL.
color	Named character vector of colors for specific characters, or NULL (default). If NULL, all characters are displayed in black. Characters specified here are automatically displayed in bold. Example: c(K = "blue", R = "red").
highlight	Named list where names are valid R colors and values are numeric vectors of positions to highlight with vertical shading. The shading spans the full height of the plot with semi-transparent color (alpha = 0.3) and black borders. Consecutive positions are automatically merged into continuous bands. To specify multiple regions with the same color, provide them as a single vector: list("#FFE0B2" = c(1:10, 50:60)). Default is NULL (no highlighting).
wrap	Numeric value specifying the maximum number of positions to display per row before wrapping to the next panel. If NULL (default), no wrapping is applied and the entire sequence is displayed in a single row. When specified, the plot is split into multiple vertically stacked panels, each showing up to wrap positions. Useful for displaying long sequences in a more compact format.
size	Numeric value for the size of sequence characters. Default is 3.
show_ref	Logical indicating whether to show the reference sequence in the plot. Default is TRUE.
margin_t	Numeric value for the top margin in points. Increases space above the plot to prevent annotation clipping. Adjust this value upward if rotated annotations are cut off. Default is 30.

`annotate` List of annotation specifications to add text labels above the plot. Each element must be a list with required elements `label` (character string) and `pos` (numeric position). Optional elements: `angle` (rotation angle), `hjust` (horizontal justification, defaults to 0 for 90-degree angles and 0.5 for other angles), `vjust` (vertical justification), `size` (text size), `face` (font face), `color` (text color). Default is NULL (no annotations).

`annotate_defaults`
List of default values for annotation styling. Valid elements: `angle` (default 0), `vjust` (default 0.5), `size` (default 3), `face` (default "plain"), `color` (default "black"). These defaults are used when individual annotations don't specify these parameters. Default is `list(angle = 0, vjust = 0.5, size = 3, face = "plain", color = "black")`.

Details

The function aligns each sequence to the reference using Needleman-Wunsch global alignment, then displays only differences. Positions matching the reference are not displayed to reduce visual clutter and emphasize mutations. Gaps from alignment are also shown as dashes.

Value

A ggplot2 object showing the aligned sequences with differences highlighted. The plot displays:

- Each sequence on a separate row
- Actual characters for positions that differ from the reference
- Optional reference sequence at the top (if `show_ref = TRUE`)
- Specified characters highlighted in bold with custom colors
- Optional vertical shading bands at specified positions
- Optional text annotations above the plot

If no valid sequences are found, returns an empty ggplot2 object with a message.

Examples

```
# ------------------------------------------------------------------------
# Example with Clustal alignment file
# ------------------------------------------------------------------------
# Create a temporary Clustal file
clustal_file <- tempfile(fileext = ".aln")
writeLines(c(
  "CLUSTAL W (1.83) multiple sequence alignment",
  "",
  "WT              EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQLERIEKKIEAHFDDLHP",
  "Mutant1         EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQLERIEKKIEAHFDDLHP",
  "Mutant2         EQKLISEEDLMKTAYIAKQRQRSFVKSHFSRQLERIEKKWEAHFDDLHP",
  "Mutant3         EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQLER----IEAHFDDLHP",
  "Mutant4         EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQAERIEKKIEAHFDDLHP",
  "Mutant5         EQKLISEEDLAKTAYIAKQRQISFVKSHFSRQLERIEKKIEAHFDDRHP",
  "Mutant6         EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQLERIEKKIEAHFDDLHP",
```

```
  "                ********** ********** * ******* *******:**",
  "",
  "WT              DIVALSGHTFGKTHGAGKQSSHHHHHH",
  "Mutant1         DIVALSGHTFGKTHGAGKQSSHHHHHH",
  "Mutant2         DIVALSGHTFGKTHGAGKQSSHHHHHH",
  "Mutant3         DIVALSGHTFGKTHGAGKQSS------",
  "Mutant4         DIVALSGHTFGKTHGAGKQSSHHHHHH",
  "Mutant5         DIVALSGHTFGKTHGAGKQSSHHHHHH",
  "Mutant6         DRVALSGHTFAKTHGAGKQSS------",
  "                * ******** **********      "
), clustal_file)

# Plot Clustal alignment
gg_seqdiff(
  clustal = clustal_file,
  ref = paste0("EQKLISEEDLMKTAYIAKQRQISFVKSHFSRQLERIEKKIEAHFDDLHP",
               "DIVALSGHTFGKTHGAGKQSSHHHHHH"),
  color = c(K = "#285bb8", R = "#285bb8",    # Basic
            E = "#a12b20", D = "#a12b20",    # Acidic
            W = "#9b59b6", F = "#9b59b6",    # Aromatic
            H = "#f39c12"),                  # Histidine
  highlight = list(
    "#94d104" = 1:10,      # N-terminal c-Myc tag
    "#FFE0B2" = 30:45,     # Active site
    "#94d104" = 72:77      # C-terminal His-tag
  ),
  annotate = list(
    list(label = "c-Myc", pos = 5),
    list(label = "Active site", pos = 37),
    list(label = "6xHis", pos = 74)
  ),
  wrap = 60
)

# Clean up
unlink(clustal_file)

# ------------------------------------------------------------------------
# Example with DNA sequences - gene structure with regulatory elements
# ------------------------------------------------------------------------
dna_ref <- paste0(
  "TATAAA",                      # TATA box (promoter)
  "ATGCGATCGATCGATCGTAGCTAGCT",   # Exon 1
  "GTAAGTATCGATCGAT",             # Intron 1 (splice sites: GT...AG)
  "ACGTACGTACGTAGCTAGCTAGCTAC",   # Exon 2
  "GTACGTACGTACGTAC",             # Intron 2
  "GTACGTACGTAGCTAGCTAGCTACGT",   # Exon 3
  "ACGTACGTAAATAA"                # 3'UTR with poly-A signal
)

dna_df <- data.frame(
  sequence = c(
    dna_ref,
```

```
    sub("TATAAA", "TATATA", dna_ref),
    gsub("GTAAGT", "ATAAGT", dna_ref),
    gsub("CGATAG", "CGATAA", dna_ref),
    sub("ATG", "AAG", dna_ref),
    gsub("AATAA$", "AACAA", dna_ref),
    sub("GCGATCGATCGATCG", "GCGATCAATCGATCG", dna_ref),
    gsub("ACGTACGTACGTAG", "ACGTACATACGTAG", dna_ref)
  ),
  id = c("WT", "Promoter_mut", "Splice_donor",
         "Splice_acceptor", "Start_codon", "PolyA_mut",
         "Exon1_missense", "Exon2_frameshift")
)

# Highlight gene structure elements
gg_seqdiff(
  data = dna_df,
  ref = dna_ref,
  name = "id",
  color = c(G = "#4e8fb5", C = "#845cab"),
  highlight = list(
    "#FFE0B2" = 1:6,                    # TATA box (promoter)
    "#C8E6C9" = c(7:32, 49:74, 91:116),  # Exons
    "#FFCCBC" = 117:130                 # 3'UTR with poly-A
  ),
  annotate = list(
    list(label = "TATA", pos = 1, angle = 90),
    list(label = "ATG", pos = 7, angle = 90, color = "red"),
    list(label = "Exon1", pos = 19),
    list(label = "GT", pos = 33, angle = 90, size = 2.5),
    list(label = "GA", pos = 46, angle = 90, size = 2.5),
    list(label = "Exon2", pos = 61),
    list(label = "GT", pos = 75, angle = 90, size = 2.5),
    list(label = "AC", pos = 89, angle = 90, size = 2.5),
    list(label = "Exon3", pos = 103),
    list(label = "AATAAA", pos = 125, angle = 90, color = "blue")
  ),
  wrap = 80
)

# ------------------------------------------------------------------------
# Example with antibody sequences with CDR mutations
# ------------------------------------------------------------------------
ref_seq <- paste0(
  "QVQLVESGGGLVQAGGSLRLSCAASGRTFSSYAMGWFRQAPGKEREFVAAINSGGSTYYP",
  "DSVKGRFTISRDNAKNTVYLQMNSLKPEDTAVYYCAADLRGTTVKDYWGQGTQVTVSSEQKLISEEDL"
)

# All sequences must be same length as reference
mutant_df <- data.frame(
  sequence = c(
    ref_seq,  # Wild-type
    # CDR1 mutations (27-33)
    sub("GRTFSSYAMG", "GRTASSYAMG", ref_seq),
```

```
    # CDR2 mutations (51-57)
    sub("AINSGGS", "AINSAGS", ref_seq),
    # CDR3 mutations (96-107)
    sub("AADLRGTTVKDY", "AADLRGTTAKDY", ref_seq),
    # Framework mutations
    sub("QVQLVES", "EVQLVAS", ref_seq),
    # Multiple CDR mutations
    sub("AADLRGTTVKDY", "AADWRGTTVKDY",
        sub("GRTFSSYAMG", "GYTASSAAMG", ref_seq))
  ),
  id = c("WT", "CDR1_F30A", "CDR2_G54A", "CDR3_V104A",
         "FR1_E5A", "CDR1+3_multi")
)

# Highlight CDRs and tags, color key residues
gg_seqdiff(
  data = mutant_df,
  ref = ref_seq,
  name = "id",
  color = c(R = "#285bb8", K = "#285bb8",  # positive
            E = "#a12b20", D = "#a12b20",  # negative
            W = "#9b59b6", F = "#9b59b6"), # aromatic
  highlight = list(
    "#70bcfa" = 1,                       # N-terminal
    "#ffb4b4" = c(27:33, 51:57, 96:107), # CDRs
    "#94d104" = 119:128                  # c-Myc tag
  ),
  annotate = list(
    list(label = "N-term", pos = 1, angle = 90),
    list(label = "CDR1", pos = 30),
    list(label = "CDR2", pos = 54),
    list(label = "CDR3", pos = 102),
    list(label = "c-Myc", pos = 123)
  ),
  wrap = 66
)
```

`gg_splitcorr` *Split-Correlation Heatmap*

Description

Creates a split-correlation plot where the upper triangle shows correlations for one subgroup and the lower triangle for another, based on a binary splitting variable. This allows quick visual comparison of correlation structures between two groups. Significant correlations (after multiple testing adjustment) can be labeled directly in the plot.

Usage

```
gg_splitcorr(
```

```
  data,
  split,
  style = "tile",
  method = "pearson",
  padjust = "BH",
  use = "complete.obs",
  colors = c("blue", "white", "red"),
  text_colors = c("white", "black"),
  text_size = 3.5,
  border_color = "black",
  prefix = NULL,
  linetype = "dashed",
  linealpha = 0.5,
  offset = 0.75
)
```

Arguments

`data`	A data frame containing numeric variables to correlate and the variable to split by.
`split`	A character string specifying the name of the binary variable in `data` used to split the dataset.
`style`	Type of visualization; either `"tile"` (default) for a heatmap or `"point"` for a bubble-style plot.
`method`	Correlation method to use; either `"pearson"` (default) or `"spearman"`.
`padjust`	Method for p-value adjustment, passed to `p.adjust` (default `"BH"`).
`use`	Handling of missing values, passed to `cor` (default `"complete.obs"`).
`colors`	A vector of three colors for the low, mid, and high values of the correlation scale (default `c("blue", "white", "red")`).
`text_colors`	A vector of two colors for the text labels, used for negative and positive correlations (default `c("white", "black")`).
`text_size`	Numeric value giving the size of correlation text labels (default `3.5`).
`border_color`	Color for tile or point borders (default `"black"`).
`prefix`	Character string prefix for group labels. If NULL (default), uses "split_variable = " format. Default is NULL.
`linetype`	Type of diagonal line separating upper and lower triangles; one of `"solid"`, `"dashed"`, or `"dotdash"` (default `"dashed"`).
`linealpha`	Alpha transparency for the diagonal line (default `0.5`).
`offset`	Numeric offset for the position of the group labels (default `0.75`).

Details

The function:

1. Splits the dataset into two groups using the variable specified by `split`.

2. Computes pairwise correlations and p-values for each group (via a helper `cor_p()`).
3. Combines the upper triangle from one group and lower triangle from the other.
4. Adjusts p-values using the selected method and annotates significant cells.

The result is a heatmap (or bubble plot) showing both groups' correlation patterns in a single compact visualization.

Value

A ggplot2 object showing the split-correlation heatmap. The plot displays:

- Upper triangle: correlations for the first level of the split variable
- Lower triangle: correlations for the second level of the split variable
- Diagonal line separating the two triangles
- Group labels indicating which split level is shown in each triangle
- Correlation values displayed only for significant pairs ($p < 0.05$ after adjustment)
- Color gradient representing correlation strength (-1 to 1)
- Optional point size (if `style = "point"`) indicating absolute correlation strength

See Also

`cor` for correlation computation, `p.adjust` for multiple testing correction, `geom_tile`, `geom_point`

Examples

```
# Compare correlations between V-shaped vs straight engines
data(mtcars)

gg_splitcorr(
  data = mtcars,
  split = "vs",
  prefix = "Engine Type: "
)

# Alternative style "point"
gg_splitcorr(
  data = mtcars,
  split = "vs",
  style = "point",
  method = "spearman",
  prefix = "Engine Type: "
)
```

Index