Package 'GPflexViz'

May 7, 2026

Title Graphical Visualizations Related to Genomic Prediction

Version 1.0.0

Description Provides flexible tools for the visualization of genomic data. Supports interactive and static plots tailored for presentations and publications, with customizable features like colors, themes, and annotations to align with specific analytical and presentation goals.

URL https://github.com/DoviniJ/GPflexViz

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports cowplot, dplyr, ggforce, ggplot2, gridExtra, nortest, plotly, reshape2, stats, utils

LazyData true

NeedsCompilation no

Author Dovini Jayasinghe [aut, cre, cph], Hong Lee [aut, cph]

Maintainer Dovini Jayasinghe <dovini.jayasinghe@mymail.unisa.edu.au>

Repository CRAN

Date/Publication 2025-09-10 07:30:14 UTC

Contents

`example_data1` *gwas_data with all chromosome information This contains Single nucleotide polymorphism (SNP) information.*

Description

gwas_data with all chromosome information This contains Single nucleotide polymorphism (SNP) information.

Usage

```
example_data1
```

Format

A dataframe with 11000 rows and 4 columns

SNP SNP ID

CHR Chromosome ID

POS Base Pair Position

P_VALUE p value

example_data10 *regression_data This contains estimation and p value data from a/the regression summary output/s.*

Description

regression_data This contains estimation and p value data from a/the regression summary output/s.

Usage

```
example_data10
```

Format

A dataframe with 9 rows and 4 columns

x_var components displayed in the horizontal axis of the heatmap

y_var components displayed in the vertical axis of the heatmap

estimate regression coefficient

p_value p value

example_data11 *CI_data This contains individual level data for constructing confidence intervals*

Description

CI_data This contains individual level data for constructing confidence intervals

Usage

```
example_data11
```

Format

A dataframe with 8 rows and 5 columns

IID Individual identification number

PRS Estimate (e.g. individual polygenic risk score (PRS))

Variance Variance of the estimate

Lower_Limit Lower limit of the confidence interval

Upper_Limit Upper limit of the confidence interval

example_data2 *gwas_data with partial chromosome information This contains Single nucleotide polymorphism (SNP) information.*

Description

gwas_data with partial chromosome information This contains Single nucleotide polymorphism (SNP) information.

Usage

```
example_data2
```

Format

A dataframe with 1000 rows and 4 columns

SNP SNP ID

CHR Chromosome ID

POS Base Pair Position

P_VALUE p value

example_data3 *accuracy_data with full information This contains prediction accuracy related information.*

Description

accuracy_data with full information This contains prediction accuracy related information.

Usage

```
example_data3
```

Format

A dataframe with 6 rows and 4 columns

Model Model name

R_squared Measuremnt of prediction accuracy

p_value p value of prediction accuracy

se standard error of prediction accuracy

`example_data4` *accuracy_data with partial information This contains prediction accuracy related information.*

Description

accuracy_data with partial information This contains prediction accuracy related information.

Usage

```
example_data4
```

Format

A dataframe with 6 rows and 2 columns

Model Model name

R_squared Measuremnt of prediction accuracy

`example_data5` *comparison_data This contains prediction accuracy related information for model comparison.*

Description

comparison_data This contains prediction accuracy related information for model comparison.

Usage

```
example_data5
```

Format

A dataframe with 15 rows and 3 columns

Compare1 Model name 1

Compare2 Model name 2

p_value p value of diffence between prediction accuracy

`example_data6` *accuracy_diff_data This contains prediction accuracy related information for detailed model comparison (2 methods at a time).*

Description

accuracy_diff_data This contains prediction accuracy related information for detailed model comparison (2 methods at a time).

Usage

```
example_data6
```

Format

A dataframe with 6 rows and 9 columns

trait Trait name

method Method name (there can be only 2 types)

R2 Prediction accuracy

lower_limit_R2 Lower limit of prediction accuracy

upper_limit_R2 Upper limit of prediction accuracy

lower_limit_difference_R2 Lower limit of prediction accuracy difference (common to the 2 methods)

upper_limit_difference_R2 Upper limit of prediction accuracy difference (common to the 2 methods)

p_value_difference_R2 p value of diffence between prediction accuracy (common to the 2 methods)

`example_data7` *ld_data This contains Linkage Disequilibrium (LD) related data (e.g. pair-wise variant correlation squared).*

Description

ld_data This contains Linkage Disequilibrium (LD) related data (e.g. pair-wise variant correlation squared).

Usage

```
example_data7
```

Format

A dataframe with 1000 rows and 7 columns

CHR_A Chromosome ID

BP_A Base pair position of the 1st variant

SNP_A First variant ID

CHR_B Chromosome ID

BP_B Base pair position of the 2nd variant

SNP_B Second variant ID

R2 LD measurement (e.g. R squared)

`example_data8` *distribution_data This contains data for consruction of histograms/density plots, with at least 1 column.*

Description

distribution_data This contains data for consruction of histograms/density plots, with at least 1 column.

Usage

```
example_data8
```

Format

A dataframe with 100 rows and 3 columns

Population_1 Data from the 1st population (e.g. phenotypes, polygenic risk scores (PRS))

Population_2 Data from the 2nd population (e.g. phenotypes, polygenic risk scores (PRS))

Population_3 Data from the 3rd population (e.g. phenotypes, polygenic risk scores (PRS))

example_data9 *correlation_matrix A 20x20 symmetric matrix containing correlation coefficients between various variables (e.g. varients, individuals).*

Description

correlation_matrix A 20x20 symmetric matrix containing correlation coefficients between various variables (e.g. varients, individuals).

Usage

```
example_data9
```

Format

A matrix with 20 rows and 20 columns with identical names for rows and columns.

Row and Column Names Both rows and columns are named VAR1 to VAR20, indicating the variables for which correlations are calculated.

flex_accuracy *flex_accuracy function*

Description

Creates a bar plot of Model prediction accuracies with the option to display p-values and make the plot interactive using Plotly.

Usage

```
flex_accuracy(
  accuracy_data,
  conf_level = 0.95,
  display_CI = TRUE,
  interactive = TRUE,
  user_colors = NULL,
  display_p_values = TRUE,
  Models_to_display_p_values = NULL,
  user_title = "Prediction Accuracy Plot",
  user_x_title = "Model",
  user_y_title = "Prediction Accuracy",
  user_legend_title = NULL,
  user_plot_theme = theme_minimal(),
 user_plot_theme_specs = theme(legend.title = element_text(size = 10), legend.text =
    element_text(size = 15), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
```

```
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  user_p_value_prefix = NULL,
  user_bar_width = 0.8,
  geom_text_args = list(),
  additional_ggplot_args = list()
)
```

Arguments

accuracy_data	A data frame containing four columns: 'Model' (factor or character vector of Model names), 'R_squared' (numeric vector of prediction accuracies), 'p_value' (numeric vector of p-values) and 'se' (numeric vector of standard errors of prediction accuracies).
conf_level	A numeric value between 0 and 1 to indicate level of confidence; default is 0.95.
display_CI	Logical; if TRUE, returns confidence intervals (CIs).
interactive	Logical; if TRUE, returns an interactive Plotly plot. If FALSE, returns a static ggplot object.
user_colors	Optional; a vector of colors to use for the bars. If NULL, default ggplot2 colors are used.
display_p_values	Logical; if TRUE, p-values are displayed on the plot.
Models_to_display_p_values	Optional; a vector of Model names for which to display p-values. If NULL, p-values are displayed for all Models.
user_title	Character string setting the title of the plot.
user_x_title	Character string for the x-axis title.
user_y_title	Character string for the y-axis title.
user_legend_title	Optional; character string for the legend title. If NULL, the legend is hidden.
user_plot_theme	ggplot2 theme object to use for the base styling of the plot.
user_plot_theme_specs	ggplot2 theme object to apply additional styling.
user_p_value_prefix	Optional; character string to prefix before p-values in their annotations.
user_bar_width	Numeric value for the width of the bars in the plot.
geom_text_args	List of additional arguments to pass to geom_text for p-value annotation customization.
additional_ggplot_args	List of additional ggplot objects to add to the plot.

Value

A ggplot object if 'interactive' is FALSE; otherwise, an interactive Plotly plot.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Usage with default bar width
flex_accuracy(example_data3)
# Example to show only specific Models' p-values with thicker bars
flex_accuracy(example_data3, display_p_values = TRUE, Models_to_display_p_values =
c("Model A", "Model E"), user_bar_width = 0.8)
# Optionally add p-value annotations
flex_accuracy(example_data3, user_p_value_prefix = "p = ", geom_text_args =
list(data = example_data3, fontface = "bold", hjust = 0.5, color = "darkblue",
size = 3))
#adding additional plotting features
flex_accuracy(example_data3, user_bar_width = 0.5, display_p_values = TRUE,
Models_to_display_p_values = c("Model A", "Model E"),
             geom_text_args = list(data = subset(example_data3,
             Model %in% c("Model A", "Model E")), angle = 60,
             fontface = "bold", hjust = 0.5, color = "darkblue", size = 5),
          additional_ggplot_args = list(ggplot2::scale_y_continuous(limits = c(0, 1.3))))
# Adjust the geom_text_args to nudge p-value annotations above the bars
flex_accuracy(example_data3,
             user_bar_width = 0.5,
             display_p_values = TRUE,
             Models_to_display_p_values = c("Model A", "Model E"),
             geom_text_args = list(angle = 60, fontface = "bold", hjust = 1,
             color = "darkblue", size = 5,
                                   nudge_y = 0.2),
          additional_ggplot_args = list(ggplot2::scale_y_continuous(limits = c(0, 1.5))))
```

`flex_accuracy_diff` *flex_accuracy_diff function*

Description

This function generates a customizable bar plot to compare prediction accuracy (e.g. R-squared values) between multiple models. It displays pairwise comparison lines and corresponding p-values. Interactive plots can be created using Plotly.

Usage

```
flex_accuracy_diff(
  accuracy_data,
```

```
  comparison_data,
  interactive = TRUE,
  user_colors = NULL,
  user_segment_color = "darkblue",
  display_p_values = TRUE,
  models_to_display_p_values = NULL,
  user_title = "Prediction Accuracy Comparison Plot",
  user_x_title = "Model",
  user_y_title = "Prediction Accuracy",
  user_legend_title = NULL,
  user_plot_theme = theme_minimal(),
 user_plot_theme_specs = theme(legend.title = element_text(size = 10), legend.text =
    element_text(size = 15), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  user_p_value_prefix = NULL,
  user_bar_width = 0.8,
  geom_text_args = list(),
  additional_ggplot_args = list()
)
```

Arguments

accuracy_data	A data frame containing the models' R-squared values with two columns: 'Model' and 'R_squared'
comparison_data	A data frame containing pairwise comparisons with three columns: 'Compare1', 'Compare2' and 'p_value'
interactive	Logical. If 'TRUE', returns an interactive plotly object; otherwise, returns a static ggplot object. Default is 'TRUE'.
user_colors	Character vector of colors for the bars. If 'NULL', uses ggplot2 default colors. Default is 'NULL'.
user_segment_color	Character. Color of comparison lines and dashed segments. Default is "darkblue"'.
display_p_values	Logical. If 'TRUE', adds p-values to the plot for specified comparisons. Default is 'TRUE'.
models_to_display_p_values	Character vector of model pairs for which p-values should be displayed, formatted as "Model1 Model2"'. Default is 'NULL'.
user_title	Character. Title of the plot. Default is "Prediction Accuracy Comparison Plot"'.
user_x_title	Character. Title of the x-axis. Default is "Model"'.
user_y_title	Character. Title of the y-axis. Default is "Prediction Accuracy"'.
user_legend_title	Character. Title of the legend. Default is 'NULL'.

user_plot_theme
: A ggplot2 theme to be applied to the plot. Default is 'theme_minimal()'.

user_plot_theme_specs
: Additional theme specifications applied after 'user_plot_theme'. Default is a set of predefined theme customizations.

user_p_value_prefix
: Character. Prefix to be added before displaying p-values. Default is 'NULL'.

user_bar_width
: Numeric. Width of the bars. Default is '0.8'.

geom_text_args
: List. Additional arguments for 'geom_text', such as color, font size, or angle. Default is an empty list.

additional_ggplot_args
: List. Additional ggplot2 layers or arguments to be added to the plot. Default is an empty list.

Value

A ggplot object if 'interactive' is FALSE; otherwise, an interactive Plotly plot.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
accuracy_data <- example_data4
comparison_data <- example_data5
# Create a static plot
flex_accuracy_diff(accuracy_data, comparison_data, interactive = FALSE)

# Create an interactive plot
flex_accuracy_diff(accuracy_data, comparison_data)

# Display specific p-values
flex_accuracy_diff(accuracy_data, comparison_data,
  models_to_display_p_values = c("Model A Model B"))
```

---

flex_accuracy_diff2 *flex_accuracy_diff2 function*

Description

This function generates detailed interactive or static visualizations comparing the accuracy of two methods across different traits. It uses ggplot2 and plotly for rendering the plots.

Usage

```
flex_accuracy_diff2(
  accuracy_diff_data,
  user_colors = c("lightblue", "lightpink"),
  interactive = TRUE,
 user_geom_bar = geom_bar(stat = "identity", position = position_dodge(0.5), width =
    0.4),
 user_geom_point = geom_point(aes(x = trait, y = R2diff), color = "black", size = 2),
  user_accuracy_geom_errorbar = geom_errorbar(aes(ymin = lower_limit_R2, ymax =
   upper_limit_R2), color = "black", width = 0.1, position = position_dodge(0.5)),
 user_accuracy_diff_geom_errorbar = geom_errorbar(width = 0.05, color = "black"),
 user_geom_text = geom_text(aes(x = trait, y = ymax + 0.1, label = p_value), hjust =
    0.5, size = 3.5),
  user_ylim_accuracy = ylim(0, 1),
  user_ylim_accuracy_difference = ylim(0, 1),
  user_accuracy_labs = labs(x = "", y = "R2", title = "R2 values"),
  user_accuracy_diff_labs = labs(x = "", y = "Difference", title =
    "Difference between R2"),
  user_accuracy_theme = theme_minimal(),
  user_accuracy_diff_theme = theme_minimal(),
  user_accuracy_theme_specs = theme(legend.position = c(0.85, 0.85)),
 user_accuracy_diff_theme_specs = theme(axis.title.x = element_blank(), axis.text.x =
   element_blank(), axis.ticks.x = element_blank(), plot.margin = unit(c(1, 1, 1, 1),
    "lines"))
)
```

Arguments

`accuracy_diff_data`
: A data frame containing nine columns and at least two rows: `trait` (factor or character vector of trait names), `method` (factor or character vector of method names (two methods)), `R2` (numeric vector of accuracy of the methods for a trait), `lower_limit_R2` (numeric vector of lower limits of accuracy), `upper_limit_R2` (numeric vector of upper limits of accuracy), `difference_R2` (numeric vector of differences between the methods' accuracies), `lower_limit_difference_R2` (numeric vector of lower limits of differences in accuracies), `upper_limit_difference_R2` (numeric vector of upper limits of differences in accuracies), and `p_value_difference_R2` (numeric vector of p-values for differences in accuracies).

`user_colors`
: A character vector of length 2 specifying colors for the two methods in the bar plot.

`interactive`
: Logical, if `TRUE` the function returns an interactive plotly plot, otherwise it returns a ggplot object.

`user_geom_bar`
: A ggplot2 geom_bar object for customizing the bar plot appearance. Accepts parameters like stat, position, and width to control the visual properties of the bars.

`user_geom_point`
: A ggplot2 geom_point object for customizing the point plot appearance. This

includes settings for color and size of points which represent the differences in accuracy.

`user_accuracy_geom_errorbar`
: A ggplot2 geom_errorbar object for customizing the error bars in the accuracy plot. This involves setting parameters like color, width, and position to visually modify how error bars are displayed.

`user_accuracy_diff_geom_errorbar`
: A ggplot2 geom_errorbar object for customizing the error bars in the accuracy difference plot. Similar to user_accuracy_geom_errorbar, but typically used to emphasize differences between methods.

`user_geom_text`
: A ggplot2 geom_text object for adding text annotations to the plots. This can include parameters for positioning, size, and the label content, often used to display statistical significance or other annotations.

`user_ylim_accuracy`
: A ggplot2 ylim function call for setting the y-axis limits in the accuracy plot. This helps in controlling the scale of the plot to better fit the data presentation.

`user_ylim_accuracy_difference`
: A ggplot2 ylim function call for setting the y-axis limits in the accuracy difference plot. Useful for maintaining consistent visual scales across related plots.

`user_accuracy_labs`
: A ggplot2 labs function call for setting labels and titles in the accuracy plot. This includes parameters to set the x-axis label, y-axis label, and the main title of the plot.

`user_accuracy_diff_labs`
: A ggplot2 labs function call for setting labels and titles in the accuracy difference plot. Useful for distinguishing between different plots and providing clear, informative titles and labels.

`user_accuracy_theme`
: A ggplot2 theme object for applying styling themes to the accuracy plot. This parameter can be used to apply a predefined theme or customize aspects like text, background, and grid lines.

`user_accuracy_diff_theme`
: A ggplot2 theme object for applying styling themes to the accuracy difference plot. Allows for consistent or contrasting styles between different types of visualizations in the package.

`user_accuracy_theme_specs`
: Additional ggplot2 theme modifications specifically for the accuracy plot. This can involve finer control over elements like legend position and plot margins.

`user_accuracy_diff_theme_specs`
: Additional ggplot2 theme modifications specifically for the accuracy difference plot. Tailored to enhance or modify specific aspects of the plot's appearance beyond the base theme settings.

Value

Either a ggplot or plotly object depending on the `interactive` argument.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
flex_accuracy_diff2(example_data6)
```

`flex_boxplot` *flex_boxplot function*

Description

This function creates a flexible, optionally interactive boxplot which can be customized with different themes, colors, and additional annotations. It supports both traditional and interactive (Plotly) outputs.

Usage

```
flex_boxplot(
  distribution_data,
  interactive = TRUE,
  user_colors = NULL,
  user_title = "Boxplot",
  user_x_title = NULL,
  user_y_title = NULL,
  user_legend_title = NA,
  user_plot_theme = theme_minimal(),
  user_plot_theme_specs = theme(legend.title = element_blank(), legend.text =
    element_text(size = 10), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  annotate_stats = FALSE,
  annotate_outliers = FALSE,
  annotate_stats_text_size = 3.5,
  annotate_outliers_text_size = 3,
  annotate_stats_text_color = "black",
  annotate_outliers_text_color = "darkred",
  annotate_outliers_text_vjust = -1,
  annotate_stats_text_vjust = 1.5,
  annotate_outliers_text_hjust = -0.5,
  annotate_stats_text_hjust = 1
)
```

Arguments

`distribution_data`
: A dataframe where each column represents a data distribution (e.g. population) to be plotted with minimum of a single column.

`interactive`
: Logical; if TRUE, the output is an interactive Plotly graph. Defaults to TRUE.

`user_colors`
: A vector of colors to be used for the boxplots. If NULL, a default set of colors is used.

`user_title`
: The main title of the plot. Defaults to "Boxplot".

`user_x_title`
: Custom title for the x-axis. If NULL, defaults to the column names of distribution_data.

`user_y_title`
: Custom title for the y-axis. If NULL, defaults to "Values".

`user_legend_title`
: Title for the legend. Use NA to hide the legend. Defaults to NA.

`user_plot_theme`
: A ggplot2 theme object to customize the appearance of the plot. Defaults to theme_minimal().

`user_plot_theme_specs`
: Additional ggplot2 theme specifications.

`annotate_stats`
: Logical; if TRUE, adds text annotations for basic statistics (min, Q1, median, Q3, max) to the plot.

`annotate_outliers`
: Logical; if TRUE, adds text annotations for outliers to the plot.

`annotate_stats_text_size`
: Numeric; text size for statistics annotations. Defaults to 3.5.

`annotate_outliers_text_size`
: Numeric; text size for outliers annotations. Defaults to 3.

`annotate_stats_text_color`
: Character; text color for statistics annotations. Defaults to "black".

`annotate_outliers_text_color`
: Character; text color for outliers annotations. Defaults to "darkred".

`annotate_outliers_text_vjust`
: Numeric; vertical adjustment for outliers text annotations.

`annotate_stats_text_vjust`
: Numeric; vertical adjustment for statistics text annotations.

`annotate_outliers_text_hjust`
: Numeric; horizontal adjustment for outliers text annotations.

`annotate_stats_text_hjust`
: Numeric; horizontal adjustment for statistics text annotations.

Value

Depending on the value of `interactive`, returns either a Plotly object and/or a ggplot object.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Basic interactive boxplot
flex_boxplot(example_data8)

# Boxplot with custom colors, themes and titles
flex_boxplot(example_data8, user_plot_theme = ggplot2::theme_gray(),
            user_colors = c("#6A3D9A", "#FF7F00", "gold1"),
            user_title = "Customized Boxplot",
          user_x_title = "Multiple Populations",
          user_y_title = "Values",
          user_legend_title = "Multiple Populations",
          user_plot_theme_specs = ggplot2::theme(
            legend.title = ggplot2::element_text(size = 10),
            legend.text = ggplot2::element_text(size = 10),
            title = ggplot2::element_text(size = 20),
            axis.text.x = ggplot2::element_blank(),
            axis.title.x = ggplot2::element_text(size = 15),
            axis.text.y = ggplot2::element_text(size = 12),
            axis.title.y = ggplot2::element_text(size = 15)
          ))
```

---

flex_correlation_plot *flex_correlation_plot function*

Description

This function creates a correlation heatmap using ggplot2 and plotly, allowing for various customizations. It supports rendering both static and interactive correlation heatmaps of either the full, lower, or upper triangular matrix.

Usage

```
flex_correlation_plot(
  correlation_matrix,
  user_colors = c("darkred", "white", "steelblue4"),
  display_names = TRUE,
  interactive = TRUE,
  user_lower_limit = -1,
  user_upper_limit = 1,
  user_mid_point = 0,
```

```
  user_plotly_x_name = "VAR_A",
  user_plotly_y_name = "VAR_B",
  user_plotly_value_name = "r",
  user_title = "Correlation Plot",
  user_x_title = NULL,
  user_y_title = NULL,
  user_legend_title = "Correlation",
  matrix_type = "full",
  user_plot_theme = theme_minimal(),
 user_plot_theme_specs = theme(legend.title = element_text(size = 10), legend.text =
    element_text(size = 10), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  user_zoom_range = NULL
)
```

Arguments

correlation_matrix	A square matrix representing correlation coefficients with column names identical to row names.
user_colors	A vector of three colors representing the gradient of the heatmap for negative, neutral, and positive correlations respectively. Default is c("darkred", "white", "steelblue4").
display_names	Logical, whether to display variable names on the axes. Default is TRUE.
interactive	Logical, indicating if the output should be an interactive plotly object. Default is TRUE.
user_lower_limit	The minimum value of the color gradient. Default is -1.
user_upper_limit	The maximum value of the color gradient. Default is 1.
user_mid_point	The midpoint value of the color gradient where the neutral color is centered. Default is 0.
user_plotly_x_name	The name to be used for the x-axis in the plotly plot. Default is "VAR_A".
user_plotly_y_name	The name to be used for the y-axis in the plotly plot. Default is "VAR_B".
user_plotly_value_name	The name to be used for the values in the plotly tooltip. Default is "r".
user_title	Title of the heatmap. Default is "Correlation Plot".
user_x_title	Custom x-axis title. If NULL, defaults to column names of the matrix.
user_y_title	Custom y-axis title. If NULL, defaults to row names of the matrix.
user_legend_title	Title for the legend. Default is "Correlation".
matrix_type	Specifies whether to plot the full matrix, the lower triangular part, or the upper triangular part. Default options are "full", "lower", "upper".

`user_plot_theme`
: ggplot2 theme object for base theming of the plot. Default is theme_minimal().

`user_plot_theme_specs`
: Additional ggplot2 theme specifications to apply on top of `user_plot_theme`.

`user_zoom_range`
: Optional numeric vector specifying the indices of the matrix to zoom into; this disables interactivity.

Value

A ggplot object if `interactive = FALSE`, otherwise a plotly interactive plot.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Generate a symmetric correlation matrix
correlation_matrix <- example_data9
# Default full matrix heatmap
flex_correlation_plot(correlation_matrix)

# Lower triangular heatmap
flex_correlation_plot(correlation_matrix, matrix_type = "lower")

# Customized plot with different theme and colors
flex_correlation_plot(
  correlation_matrix = correlation_matrix,
  user_colors = c("gold2", "lightgrey", "darkblue"),
  user_title = "Custom Correlation Plot",
  user_x_title = "Variables",
  user_y_title = "Variables",
  user_legend_title = "Correlation Coefficient",
  user_plot_theme = theme_classic(),
  user_plot_theme_specs = theme(
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10),
    title = element_text(size = 16),
    axis.title.x = element_text(size = 12),
    axis.text.x = element_text(size = 12, angle = 45, hjust = 1),
    axis.text.y = element_text(size = 12),
    axis.title.y = element_text(size = 12),
    legend.position = "bottom")
)
```

`flex_distribution` *flex_distribution function*

Description

The `flex_distribution` function provides a flexible way to visualize data distributions using histograms and density plots. It can be used to create both basic and interactive plots with additional features such as density curves, reference lines, and summary statistics annotations. The function supports custom colors and various themes.

Usage

```
flex_distribution(
  distribution_data,
  interactive = TRUE,
  user_colors = NULL,
  plot_type = "histogram",
  add_density = FALSE,
  reference_line = "mean",
  show_summary = FALSE,
  user_title = "Distribution Plot",
  user_x_title = NULL,
  user_y_title = "Frequency",
  user_legend_title = NA,
  user_plot_theme = theme_minimal(),
  user_plot_theme_specs = theme(legend.title = element_blank(), legend.text =
    element_text(size = 10), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  binwidth = NULL,
  bins = NULL
)
```

Arguments

`distribution_data`	A dataframe where each column represents a data distribution (e.g. population) to be plotted with minimum of a single column.
`interactive`	Logical, if TRUE (default), the plot will be interactive using Plotly, otherwise a ggplot object is returned.
`user_colors`	A vector of colors for the plots. If NULL, a set of default colors is used.
`plot_type`	The type of plot to create, with options "histogram" and/or "density".
`add_density`	Logical, if TRUE, adds a density curve to the plot.
`reference_line`	Specifies the type of reference line to add; options include "mean", "median", or FALSE for no line.

show_summary	Logical, if TRUE, displays summary statistics on the plot.
user_title	Title of the plot.
user_x_title	Custom X-axis title. If NULL, the name of the variable is used.
user_y_title	Custom Y-axis title, default is "Frequency".
user_legend_title	Title for the legend. Can be NA to exclude the legend title.
user_plot_theme	ggplot2 theme object for customizing the appearance of the plot.
user_plot_theme_specs	Further ggplot2 theme specifications.
binwidth	The width of the bins for the histogram (optional).
bins	The number of bins for the histogram (optional).

Details

The function is designed to be flexible and allows extensive customization of plot aesthetics and functionality. Density plots are added as an additional layer when add_density is TRUE, and the reference line can be specified by the user.

Value

An object of class 'ggplot' or 'plotly' depending on the 'interactive' parameter.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Histogram with custom colors
custom_colors <- c("red", "blue", "green")
flex_distribution(example_data8, user_colors = custom_colors)

# Density plot
flex_distribution(
  example_data8,
  plot_type = "density",
  user_colors = custom_colors,
  user_y_title = "Density"
)

#' # Basic histogram with density curves and summary statistics
flex_distribution(
  example_data8,
```

```
  add_density = TRUE,
  show_summary = TRUE,
  user_title = "Distribution with Densities and Summary",
  user_y_title = "Density"
)

# Histogram with custom binwidth
flex_distribution(
  example_data8,
  binwidth = 0.5,
  user_title = "Distribution with Custom Binwidth"
)

# Histogram with a specified number of bins
flex_distribution(
  example_data8,
  bins = 50,
  user_title = "Distribution with Custom Bin Count"
)
```

`flex_interval` *flex_interval function*

Description

This function visualizes individual polygenic risk scores (PRS) with corresponding confidence intervals. It supports customization of reference lines, colors, themes, and interactivity.

Usage

```
flex_interval(
  CI_data,
  user_ref_line = NULL,
  user_ref_color = "red",
  user_ref_linetype = "dashed",
  user_ref_visible = TRUE,
  user_point_color = "blue",
  user_point_size = 3,
  user_errorbar_color = "black",
  user_errorbar_size = 0.2,
  interactive = TRUE,
  user_theme = theme_minimal(),
  user_theme_specifications = theme(),
  user_title = "PRS with Confidence Intervals",
  user_x_label = "Individual",
  user_y_label = "Polygenic Risk Score (PRS)"
)
```

Arguments

CI_data	Data frame containing columns: IID, PRS, Variance, Lower_Limit, and Upper_Limit.
user_ref_line	Numeric or vector. Horizontal reference line(s) to be added. If NULL, defaults to mean PRS. Default is NULL.
user_ref_color	Character or vector. Color(s) of reference line(s). Default is "red".
user_ref_linetype	Character or vector. Linetype(s) for reference line(s). Default is "dashed".
user_ref_visible	Logical. Whether to display the reference line(s). Default is TRUE.
user_point_color	Character. Color of PRS points. Default is "blue".
user_point_size	Numeric. Size of PRS points. Default is 3.
user_errorbar_color	Character. Color of confidence interval error bars. Default is "black".
user_errorbar_size	Numeric. Width of error bars. Default is 0.2.
interactive	Logical. If TRUE, returns an interactive plot using plotly. Default is TRUE.
user_theme	A ggplot2 theme object. Default is theme_minimal().
user_theme_specifications	Additional theme specifications. Default is an empty theme() object.
user_title	Character. Title of the plot. Default is "PRS with Confidence Intervals".
user_x_label	Character. X-axis label. Default is "Individual".
user_y_label	Character. Y-axis label. Default is "Polygenic Risk Score (PRS)".

Value

A ggplot object (static plot) or a plotly object (interactive plot), depending on the value of interactive.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
set.seed(123)
example_data <- data.frame(
  IID = paste0("ID_", 1:8),
  PRS = rnorm(8, 0, 1),
  Variance = runif(8, 0.01, 0.1)
)
```

```
example_data$Lower_Limit <- example_data$PRS - 1.96 * sqrt(example_data$Variance)
example_data$Upper_Limit <- example_data$PRS + 1.96 * sqrt(example_data$Variance)
CI_data <- example_data

# Basic
flex_interval(CI_data)

# Add custom reference lines
flex_interval(CI_data,
              user_ref_line = c(1, -1),
              user_ref_color = c("darkgreen", "darkred"),
              user_ref_linetype = c("dotted", "twodash"))

# Static plot with theme customization
flex_interval(CI_data, interactive = FALSE,
         user_theme_specifications = theme(axis.text.x = element_text(angle = 45, hjust = 1)),
              user_point_color = "magenta",
              user_errorbar_color = "darkred")

# Sorted PRS plot
ordered_data <- CI_data[order(-CI_data$PRS),]
ordered_data$IID <- factor(ordered_data$IID, levels = ordered_data$IID)
flex_interval(ordered_data, interactive = FALSE, user_ref_visible = FALSE)
```

`flex_LD_decay` *flex_LD_decay function*

Description

This function generates a linkage disequilibrium (LD) decay plot based on input LD data. It supports both static and interactive visualization with customizable aesthetics, themes, and optional smoothing curves.

Usage

```
flex_LD_decay(
  ld_data,
  interactive = TRUE,
  user_colors = NULL,
  user_title = "LD Decay Plot",
  user_x_title = "Distance (Mb)",
  user_y_title = "R-squared (LD)",
  user_legend_title = "Chromosome",
  user_plot_theme = theme_minimal(),
 user_plot_theme_specs = theme(legend.title = element_text(size = 10), legend.text =
    element_text(size = 10), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  user_base = 1e+06,
```

```
  user_smoothing = "loess",
  add_smoothing = FALSE,
  add_smoothing_per_chromosome = FALSE,
  ...
)
```

Arguments

ld_data	Data frame containing LD information.
interactive	Logical. If TRUE, returns an interactive plot using plotly. Default is TRUE.
user_colors	Character vector of colors for chromosomes. If NULL, default colors are used. Default is NULL.
user_title	Character. Title of the plot. Default is "LD Decay Plot".
user_x_title	Character. X-axis label. Default is "Distance (Mb)".
user_y_title	Character. Y-axis label. Default is "R-squared (LD)".
user_legend_title	Character. Legend title. Default is "Chromosome".
user_plot_theme	A ggplot2 theme object. Default is theme_minimal().
user_plot_theme_specs	Additional theme specifications applied to the plot. Default is a theme object with custom font sizes.
user_base	Numeric. Scaling factor for distance (e.g., 1e6 for Mb). Default is 1e6.
user_smoothing	Character. Smoothing method to use in geom_smooth. Default is "loess".
add_smoothing	Logical. If TRUE, adds a smoothing curve to the plot. Default is FALSE.
add_smoothing_per_chromosome	Logical. If TRUE, adds separate smoothing curves for each chromosome. Default is FALSE.
...	Additional arguments passed to geom_smooth (e.g., span, se, etc.).

Value

A ggplot object (static plot) or a plotly object (interactive plot), depending on the value of interactive.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Load example data
```

```
snp_ld_data <- example_data7

# Basic plot
flex_LD_decay(snp_ld_data)

# Add a general smoothing curve
flex_LD_decay(snp_ld_data, add_smoothing = TRUE)

# Add smoothing per chromosome
flex_LD_decay(snp_ld_data, add_smoothing = TRUE, add_smoothing_per_chromosome = TRUE)

# Customize smoothing parameters
flex_LD_decay(snp_ld_data, add_smoothing = TRUE, span = 0.5, se = FALSE)

# Use generalized additive model (GAM) for smoothing
flex_LD_decay(snp_ld_data, user_smoothing = "gam", add_smoothing = TRUE,
add_smoothing_per_chromosome = TRUE)
```

`flex_manhattan` *flex_manhattan function*

Description

This function creates a flexible Manhattan plot, which is useful for visualizing GWAS results. The plot can be generated as either a static or an interactive plot. The function allows customization of color schemes, titles, axis labels, and annotation features.

Usage

```
flex_manhattan(
  gwas_data,
  interactive = TRUE,
  user_colors = NULL,
  user_y_cutoff = -log10(5e-08),
  user_y_cutoff_color = "red",
  user_title = "Manhattan Plot",
  user_x_title = "Position",
  user_y_title = "-log10(p-value)",
  user_legend_title = "Chromosome",
  user_plot_theme = theme_minimal(),
 user_plot_theme_specs = theme(legend.title = element_text(size = 10), legend.text =
    element_text(size = 10), title = element_text(size = 15), axis.text.x =
   element_text(size = 10), axis.title.x = element_text(size = 10), axis.text.y =
    element_text(size = 10), axis.title.y = element_text(size = 10)),
  annotate_data = NULL,
  annotate_column = "SNP",
  annotate_labels = FALSE,
  zoom_on_annotations = FALSE,
```

```
  zoom_margin = 1,
  ...
)
```

Arguments

gwas_data	A data frame containing columns for SNP ID ('SNP'), chromosome ('CHR'), position ('POS'), and p-values ('P_VALUE').
interactive	Logical; if TRUE, returns an interactive plotly plot, otherwise a ggplot2 plot.
user_colors	A vector of colors for the chromosomes. If NULL, a default set of 22 colors is used.
user_y_cutoff	A numeric value for the y-axis cutoff, used to draw a horizontal line.
user_y_cutoff_color	Color of the y-axis cutoff line.
user_title	The main title of the plot.
user_x_title	The label for the x-axis.
user_y_title	The label for the y-axis.
user_legend_title	The title for the legend. If NULL, no legend is displayed.
user_plot_theme	ggplot2 theme object for styling the plot background and fonts.
user_plot_theme_specs	Additional ggplot2 theme specifications for custom styling.
annotate_data	A vector of SNP identifiers for which annotations are to be made.
annotate_column	The column name from gwas_data used for matching annotate_data.
annotate_labels	Logical; if TRUE, annotations are added to the plot.
zoom_on_annotations	Logical; if TRUE, the plot will zoom in on annotated SNPs.
zoom_margin	Numeric; defines the margin around the zoomed area as a proportion of the range of the data.
...	Additional arguments to be passed to ggplot2 plotting functions.

Value

An object of class 'ggplot' or 'plotly' depending on the 'interactive' parameter.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
```

```
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
flex_manhattan(example_data1)

# Changing the y-axis cutoff to highlight significant p-values differently
flex_manhattan(example_data1,
               user_y_cutoff = c(5, 4),  # Multiple cutoffs
               user_y_cutoff_color = c("darkred", "darkgreen"))  # Corresponding colors

# Adding annotations for specific SNPs
annotated_snps <- c("SNP_2550", "SNP_4829", "SNP_8296")
flex_manhattan(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE,
interactive = FALSE)


# Example of placing the legend at the bottom with a black outline
flex_manhattan(example_data1,
               user_plot_theme_specs = theme(
                 legend.position = "bottom",
                 legend.box.background = element_rect(colour = "black")
               ))
```

`flex_qqplot` *flex_qqplot function*

Description

This function generates a customizable quantile-quantile (QQ) plot for GWAS data, allowing for interactive exploration, zooming on specific SNPs, and inclusion of a Kolmogorov-Smirnov test result annotation.

Usage

```
flex_qqplot(
  gwas_data,
  interactive = TRUE,
  display_ks = FALSE,
  user_colors = c("dodgerblue", "darkorange"),
  user_title = "QQ Plot",
  user_x_title = "Theoretical Quantiles",
  user_y_title = "Observed Quantiles",
  user_plot_theme = theme_minimal(),
  user_plot_theme_specs = theme(title = element_text(size = 15), axis.text =
    element_text(size = 10), axis.title = element_text(size = 10)),
  annotate_data = NULL,
  annotate_column = "SNP",
```

```
  annotate_labels = FALSE,
  zoom_on_annotations = FALSE,
  zoom_margin = 1,
  ...
)
```

Arguments

gwas_data	A data frame containing columns for SNP ID ('SNP'), chromosome ('CHR'), position ('POS'), and p-values ('P_VALUE').
interactive	Logical, whether to return an interactive plot (TRUE) or static ggplot (FALSE).
display_ks	Logical, whether to display the Kolmogorov-Smirnov (KS) test result.
user_colors	Character vector of length 2 specifying the colors for points and reference line.
user_title	The title of the plot.
user_x_title	The title for the x-axis.
user_y_title	The title for the y-axis.
user_plot_theme	A ggplot2 theme object to style the plot.
user_plot_theme_specs	Additional ggplot2 theme specifications to override in user_plot_theme.
annotate_data	A vector of SNP identifiers to annotate in the plot.
annotate_column	The name of the column in gwas_data to use for matching annotate_data.
annotate_labels	Logical, whether to label the annotated points.
zoom_on_annotations	Logical, whether to create a zoomed version of the plot focusing on annotated SNPs.
zoom_margin	Numeric, the margin size around zoomed points.
...	Additional arguments to be passed to ggplot2 plotting functions.

Value

An object of class 'ggplot' or 'plotly' depending on the 'interactive' parameter.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
```

```
flex_qqplot(example_data1, user_colors = c("black", "darkorange"))
flex_qqplot(example_data1, display_ks = TRUE, interactive = FALSE)
annotated_snps <- c("SNP_1", "SNP_2", "SNP_3")
flex_qqplot(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE)
flex_qqplot(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE,
display_ks = TRUE)
flex_qqplot(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE,
zoom_on_annotations = TRUE, zoom_margin = 0.3, user_plot_theme = theme_bw())
```

`flex_regression_summary`

flex_regression_summary function

Description

This function creates a flexible regression summary plot with customization options for interactivity, p-value intervals, legend sizes, color gradients, and more. It supports both ggplot2 and plotly outputs for static and interactive visualizations, respectively.

Usage

```
flex_regression_summary(
  regression_data,
  interactive = TRUE,
 user_size_breaks = cut(regression_data$p_value, breaks = c(0, 0.01, 0.05, 0.1, 1),
    labels = c("< 0.01", "0.05", "0.1", "1"), include.lowest = TRUE),
  user_legend_break_sizes = c(12, 10, 8, 7),
  user_geom_tile = geom_tile(color = "white", fill = "white", size = 5),
  user_geom_point = geom_point(shape = 22, color = "white", stroke = 0),
 user_gradient_bar = scale_fill_gradientn(colours = c("red4", "white", "steelblue4"),
    limit = c(-max(abs(regression_data$estimate)) - 0.05,
    max(abs(regression_data$estimate)) + 0.05), guide = guide_colorbar(title =
   "Estimate", label.position = "right", barwidth = 1, barheight = 5, title.position =
    "top", order = 1)),
 user_geom_text = list(geom_text(data = subset(regression_data, (p_value <= 0.01)),
  aes(label = "**"), size = 6, vjust = 0.5, hjust = 0.5, show.legend = FALSE, color =
  "white", fontface = "bold"), geom_text(data = subset(regression_data, (p_value <=
   0.05 & p_value > 0.01)), aes(label = "*"), size = 6, vjust = 0.5, hjust = 0.5,
    show.legend = FALSE, color = "white", fontface = "bold")),
 user_theme = theme_minimal(),
 user_theme_specs = theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1,
  size = 15, colour = "black"), axis.text.y = element_text(size = 15), axis.title.y =
    element_text(size = 15), axis.title.x = element_text(size = 15), text =
  element_text(family = "serif"), legend.text = element_text(size = 15), legend.title =
  element_text(size = 15), legend.key.width = unit(0.01, "cm"), legend.key.height =
  unit(0.01, "cm"), legend.box = "horizontal", legend.spacing.y = unit(0.5, "cm"),
    legend.position = "right", legend.box.margin = margin(b = 15,
```

```
    t = -20, l = 5, r
  = 5), legend.box.just = ("bottom"), plot.margin = margin(b = 5, t = 20, l = 5, r =
    5)),
 user_labs = labs(x = "x variable", y = "y variable", fill = "Estimate", size =
   "p value"),
 user_tooltip = list("Estimate: ", "p-value: "),
user_size_legend_guides = guides(size = guide_legend(reverse = FALSE, label.position =
 "right", title.position = "top", override.aes = list(colour = "grey", shape = 15,
   fill = "white", size = user_legend_break_sizes - 3)))
)
```

Arguments

`regression_data`	A dataframe containing the variables `x_var`, `y_var`, `estimate`, and `p_value` which represent the horizontal axis component, vertical axis component, regression coefficient estimates, and their corresponding p-values, respectively.
`interactive`	Logical, if TRUE returns an interactive plotly object, otherwise returns a ggplot object.
`user_size_breaks`	Factor levels for breaking down the p-value into intervals. Default is predefined intervals at $p < 0.01$, 0.05, 0.1, and 1.
`user_legend_break_sizes`	A numeric vector indicating the sizes of points in the legend, corresponding to the p-value intervals.
`user_geom_tile`	Custom ggplot2 geom_tile layer, allows customization of tile appearance.
`user_geom_point`	Custom ggplot2 geom_point layer, allows customization of point markers.
`user_gradient_bar`	Custom ggplot2 scale_fill_gradientn for coloring the tiles based on the estimates.
`user_geom_text`	List of custom ggplot2 geom_text layers for adding text annotations based on significance.
`user_theme`	Custom ggplot2 theme, allows overriding the default minimal theme.
`user_theme_specs`	Custom modifications to the default or user-specified theme.
`user_labs`	Custom ggplot2 labs function for setting axis and legend titles.
`user_tooltip`	A list containing tooltip strings for interactive plots.
`user_size_legend_guides`	Custom ggplot2 guides for the size aesthetic in the legend.

Value

A ggplot or plotly object depending on the `interactive` argument.

Examples

```
library(ggplot2)
library(plotly)
library(dplyr)
library(nortest)
library(ggforce)
library(reshape2)
library(gridExtra)
library(grid)
library(cowplot)
# Default usage
flex_regression_summary(example_data10)

# Custom p-value intervals and annotations
flex_regression_summary(example_data10, user_size_breaks = cut(example_data10$p_value,
                 breaks = c(0, 0.05, 1), labels = c("< 0.05", "1"), include.lowest = TRUE),
                        user_legend_break_sizes = c(12, 8),
                        user_geom_text = list(geom_text(data = subset(example_data10,
                                                (p_value <= 0.05)),
                                aes(label = "sig"),
                                size = 4, vjust = 0.5, hjust = 0.5,
                                color = "black",
                                fontface = "italic")))

# Custom formatting and gradient colors
flex_regression_summary(example_data10, interactive = FALSE,
user_geom_point = geom_point(shape = 25,
color = "white", stroke = 0),
                       user_gradient_bar = scale_fill_gradientn(colours = c("purple4",
                       "white", "green4")))
```

Index