User Manual

R package: **GPflexViz**
Version: 1.0.0

Contents

1 General Description

GPflexViz (Genomic Prediction Flexible Visualization) is a versatile, user-friendly, and comprehensive R package designed to empower users with flexible visualization tools for genomic data analysis. This all-in-one solution caters to a wide range of needs, from interactive plots that elevate presentations to static, publication-ready visuals. Unlike one-size-fits-all tools, GPflexViz prioritizes customizability, enabling users to tailor every aspect of their visualizations—colors, themes, annotations, and more—ensuring that plots are not only visually appealing but also aligned with specific analytical and presentation goals. With intuitive functions, GPflexViz simplifies complex genomic data exploration, making it accessible for researchers at any level of expertise. Whether for interpretation, reporting, or public dissemination, this package is designed to adapt, delivering insights that are both engaging and impactful.

2 Dependencies

Users are required to install and load the following libraries:

```
install.packages(c("ggplot2", "plotly", "dplyr", "nortest", "ggforce", "reshape2",
"htmlwidgets", "gridExtra", "grid", "cowplot"))

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

Important

Users are recommended to use RStudio for generating/saving interactive plots. To install RStudio, click on, `https://posit.co/download/rstudio-desktop/` and follow the instructions.

Install and load the following library to save interactive plots as .html files:

```
install.packages("htmlwidgets")
library(htmlwidgets)
```

However, static plots can be generated/saved (e.g. as .png file) in any platform (R or RStudio).

3 Installation of GPflexViz

GitHub installation:

```
install.packages("devtools")
devtools::install_github("DoviniJ/GPflexViz")
```

Then load the library:

```
library(GPflexViz)
```

4 Input File Formats

4.1 gwas_data

A data frame containing columns for SNP ID (SNP), chromosome ('CHR'), position ('POS'), and p-values ('P_VALUE').

```
> head(example_data1) #example dataframe with all 22 chromosomes
    SNP CHR     POS    P_VALUE
1 SNP_1   1 1000261 0.08736310
2 SNP_2   1 1000375 0.41315718
3 SNP_3   1 1001861 0.09806681
4 SNP_4   1 1002499 0.91615572
5 SNP_5   1 1002853 0.06461653
6 SNP_6   1 1003304 0.38995607

> head(example_data2) #example dataframe with first 5 chromosomes
    SNP CHR     POS   P_VALUE
1 SNP_1   1 1016443 0.2883334
2 SNP_2   1 1016486 0.4107970
3 SNP_3   1 1017130 0.5499115
4 SNP_4   1 1017959 0.4405514
5 SNP_5   1 1022432 0.9652469
6 SNP_6   1 1023848 0.2763288
```

4.2 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).

Note: Users may obtain the R squared specific values from the R package "r2redux" `https://github.com/mommy003/r2redux/tree/main`. If the accuracy is measured in terms of AUC values, the same could be computed using "R2ROC" `https://github.com/mommy003/R2ROC` R package.

```
> head(example_data3)
    Model R_squared p_value    se
1 Model A      0.95 0.00810 0.010
2 Model B      0.89 0.06548 0.020
3 Model C      0.78 0.03782 0.015
4 Model D      0.65 0.00545 0.030
5 Model E      0.29 0.09453 0.025
6 Model F      0.78 0.03450 0.020

> head(example_data4) #a subset of accuracy_data containing only Model and R_squared columns
    Model R_squared
1 Model A      0.95
2 Model B      0.89
3 Model C      0.78
4 Model D      0.65
5 Model E      0.29
6 Model F      0.78
```

4.3 comparison_data

A data frame containing three columns: 'Compare1' (factor or character vector of model names), 'Compare2' (factor or character vector of model names) and 'p_value' (numeric vector of p-values for the difference between prediction accuracy).

```
> head(example_data5)
  Compare1 Compare2 p_value
1  Model A  Model B    0.04
```

```
2  Model A  Model C    0.07
3  Model A  Model D    0.01
4  Model A  Model E    0.15
5  Model A  Model F    0.03
6  Model B  Model C    0.05
```

4.4 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' (accuracy of the first/second method, given a trait), 'lower_limit_R2' (lower limit of accuracy value), 'upper_limit_R2' (upper limit of accuracy value), 'difference_R2' (difference between accuracy values of the methods, given a trait), 'lower_limit_difference_R2' (lower limit of difference of the accuracy value), 'upper_limi_difference_R2' (upper limit of difference of the accuracy value) and 'p_value_difference_R2' (numeric vector of p-values of difference of the accuracy value).

Note: Users may obtain the R squared specific values from the R package "r2redux" `https://github.com/mommy003/r2redux/tree/main`. If the accuracy is measured in terms of AUC values, the same could be computed using "R2ROC" `https://github.com/mommy003/R2ROC` R package.

```
> head(example_data6)
    trait   method   R2 lower_limit_R2 upper_limit_R2 difference_R2
1 Triat 1 Method 1 0.40          0.300          0.500          0.40
2 Triat 1 Method 2 0.80          0.750          0.850          0.40
3 Triat 2 Method 1 0.60          0.300          0.900          0.20
4 Triat 2 Method 2 0.80          0.650          1.000          0.20
5 Triat 3 Method 1 0.26          0.230          0.290          0.02
6 Triat 3 Method 2 0.28          0.265          0.295          0.02
  lower_limit_difference_R2 upper_limit_difference_R2 p_value_difference_R2
1                     0.320                     0.520              3.60E-09
2                     0.320                     0.520              3.60E-09
3                     0.140                     0.340             2.060E-02
4                     0.140                     0.340             2.060E-02
5                     0.014                     0.034              8.60E-01
6                     0.014                     0.034              8.60E-01
```

4.5 ld_data

A data frame containing LD information. Required columns include:

'CHR_A' - Chromosome identifier for SNP_A.

'BP_A' - Base pair position of SNP_A.

'SNP_A' - Identifier for SNP_A.

'CHR_B' - Chromosome identifier for SNP_B.

'BP_B' - Base pair position of SNP_B.

'SNP_B' - Identifier for SNP_B.

'R2' - Linkage disequilibrium (R-squared) value.

```
> head(example_data7)
  CHR_A      BP_A SNP_A CHR_B      BP_B SNP_B       R2
1     3  11549596 SNP_1     3  11593013 SNP_2 0.790378
2    19  41248009 SNP_2    19  41281016 SNP_3 0.653171
3    12  11188140 SNP_3    12  11338781 SNP_4 0.203153
```

```
4    12  89468283 SNP_4    12  89520000 SNP_5 0.211142
5     4  32988370 SNP_5     4  33150243 SNP_6 0.201035
6     5 137834139 SNP_6     5 137883734 SNP_7 0.308624
```

4.6 distribution_data

A data frame containing at least one column (e.g. PRSs of (a certain) population/s).

```
> head(example_data8)
  Population_1 Population_2 Population_3
1  -0.56047565    0.7561026   -1.1310259
2  -0.23017749    1.5465069    0.8194890
3   1.55870831    3.2864430   -1.1393340
4   0.07050839    0.8939267    1.2557349
5   0.12928774    1.4453594   -0.7689445
6   1.71506499    2.1047045    0.7509707
```

4.7 correlation_matrix

A square matrix of correlation values with identical row and column names.

```
> head(example_data9)
      VAR 1 VAR 2 VAR 3 VAR 4 VAR 5 VAR 6 VAR 7 VAR 8 VAR 9 VAR 10 VAR 11 VAR 12
VAR 1  1.00 -0.10 -0.34 -0.02  0.32 -0.28 -0.09 -0.40 -0.15   0.34   0.25  -0.08
VAR 2 -0.10  1.00  0.11 -0.05  0.24 -0.21  0.07  0.16 -0.24   0.20  -0.06   0.40
VAR 3 -0.34  0.11  1.00 -0.17 -0.20  0.07 -0.01 -0.20  0.25   0.08  -0.17   0.18
VAR 4 -0.02 -0.05 -0.17  1.00 -0.09  0.26  0.43 -0.03 -0.15   0.19  -0.11  -0.33
VAR 5  0.32  0.24 -0.20 -0.09  1.00 -0.02 -0.15 -0.18  0.02  -0.19   0.27  -0.08
VAR 6 -0.28 -0.21  0.07  0.26 -0.02  1.00  0.19 -0.25  0.29  -0.17  -0.03  -0.08
      VAR 13 VAR 14 VAR 15 VAR 16 VAR 17 VAR 18 VAR 19 VAR 20
VAR 1  -0.09  -0.01  -0.30  -0.21   0.43  -0.22  -0.34  -0.28
VAR 2  -0.04  -0.28  -0.05  -0.09  -0.26   0.36  -0.07  -0.08
VAR 3  -0.19  -0.11   0.09  -0.13   0.19  -0.28   0.08   0.46
VAR 4   0.09   0.15  -0.12  -0.09  -0.42   0.14  -0.12   0.10
VAR 5   0.03  -0.28   0.22   0.12  -0.04   0.22  -0.05  -0.11
VAR 6   0.34   0.16   0.11   0.44  -0.06  -0.19   0.10   0.40
```

4.8 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.

```
> head(example_data10)
  x_var y_var estimate p_value
1    x1    y1     -0.2  0.8569
2    x1    y2      0.5  0.0450
3    x1    y3      0.8  0.0020
4    x2    y1     -0.3  0.0852
5    x2    y2     -0.7  0.0550
6    x2    y3     -0.5  0.0650
```

5 Functions

5.1 flex_manhattan()

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.

5.1.1 Arguments

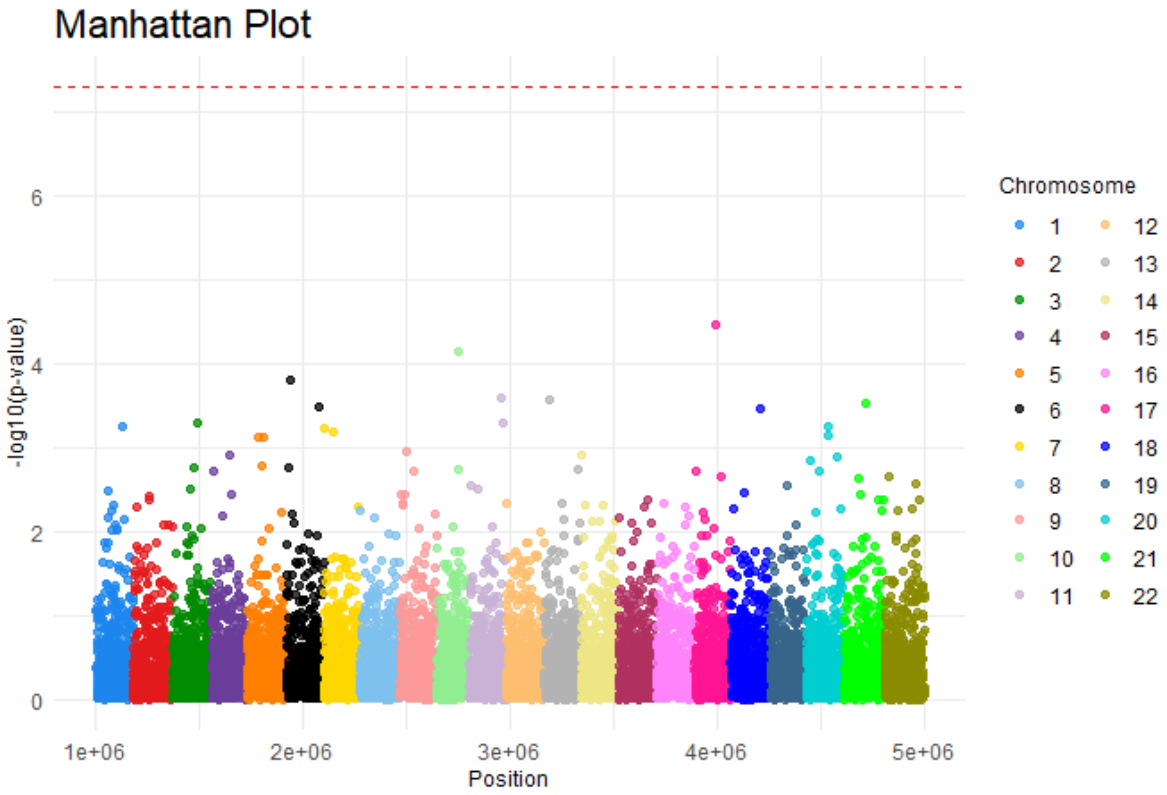
- gwas_data: A data frame containing columns for SNP ID ('SNP'), chromosome ('CHR'), position ('POS'), and p-values ('P_VALUE').
- interactive: Logical; if TRUE, returns an interactive plotly plot, otherwise a ggplot2 plot.
- user_colors: A vector of colors for the chromosomes. If NULL, a default set 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.

5.1.2 Default manhattan plot

Command:

```
flex_manhattan(example_data1)
```

Output:

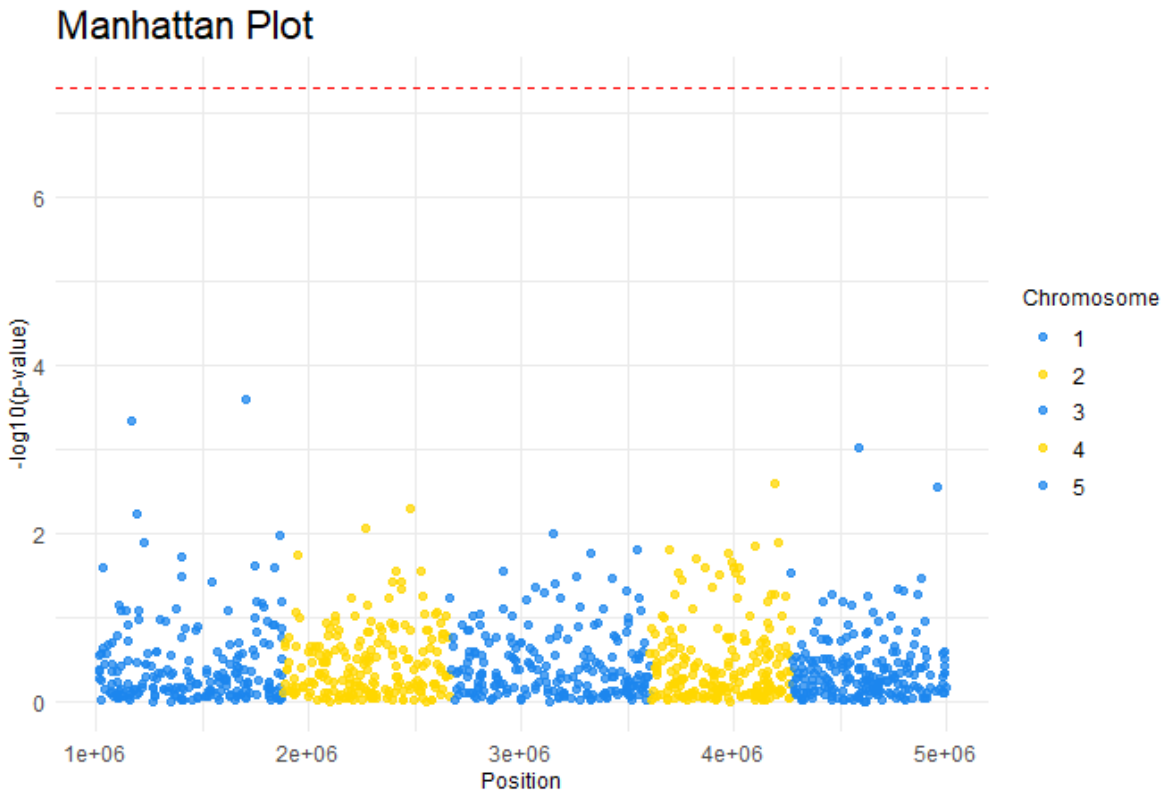


5.1.3 Manhattan plot with user-specified colors

Command:

```
flex_manhattan(example_data2,
               user_colors = rep(c("dodgerblue2","gold1"),
                                 length.out = max(unique(example_data2$CHR))))
```

Output:

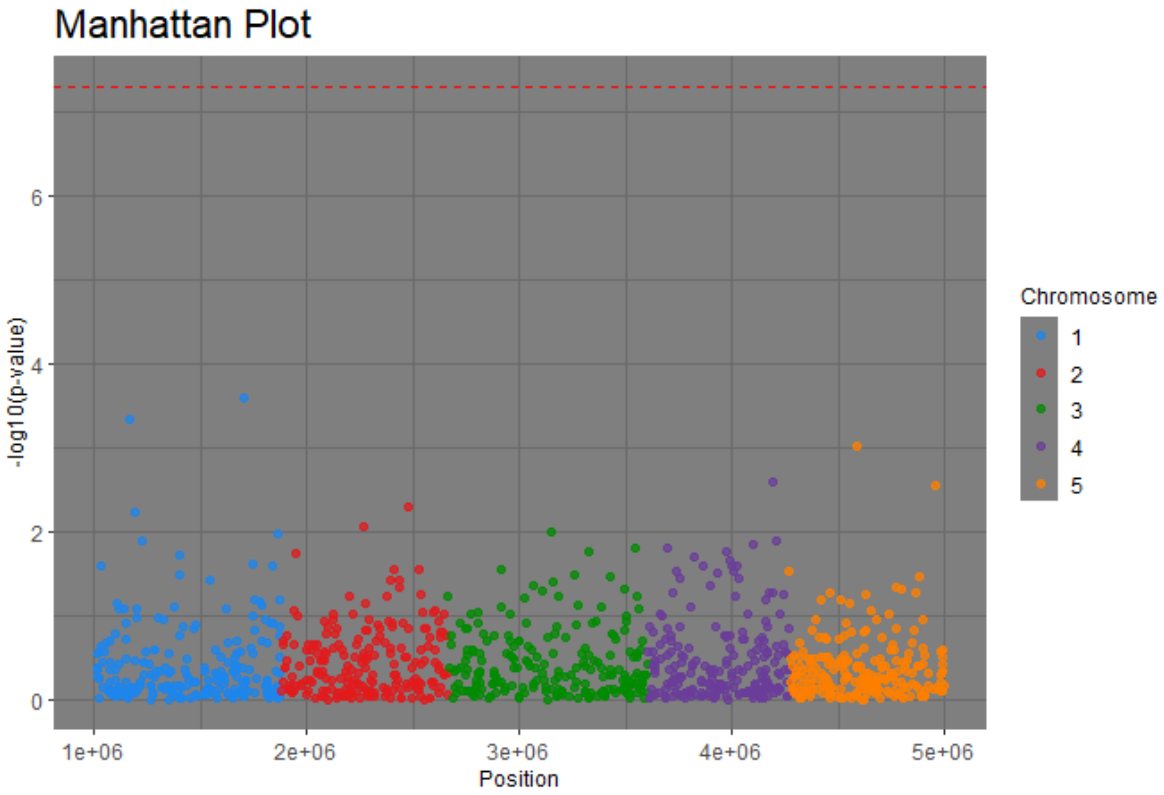


5.1.4 Manhattan plot with user-specified theme type

Command:

```
flex_manhattan(example_data2, user_plot_theme = theme_dark())
```

Output:

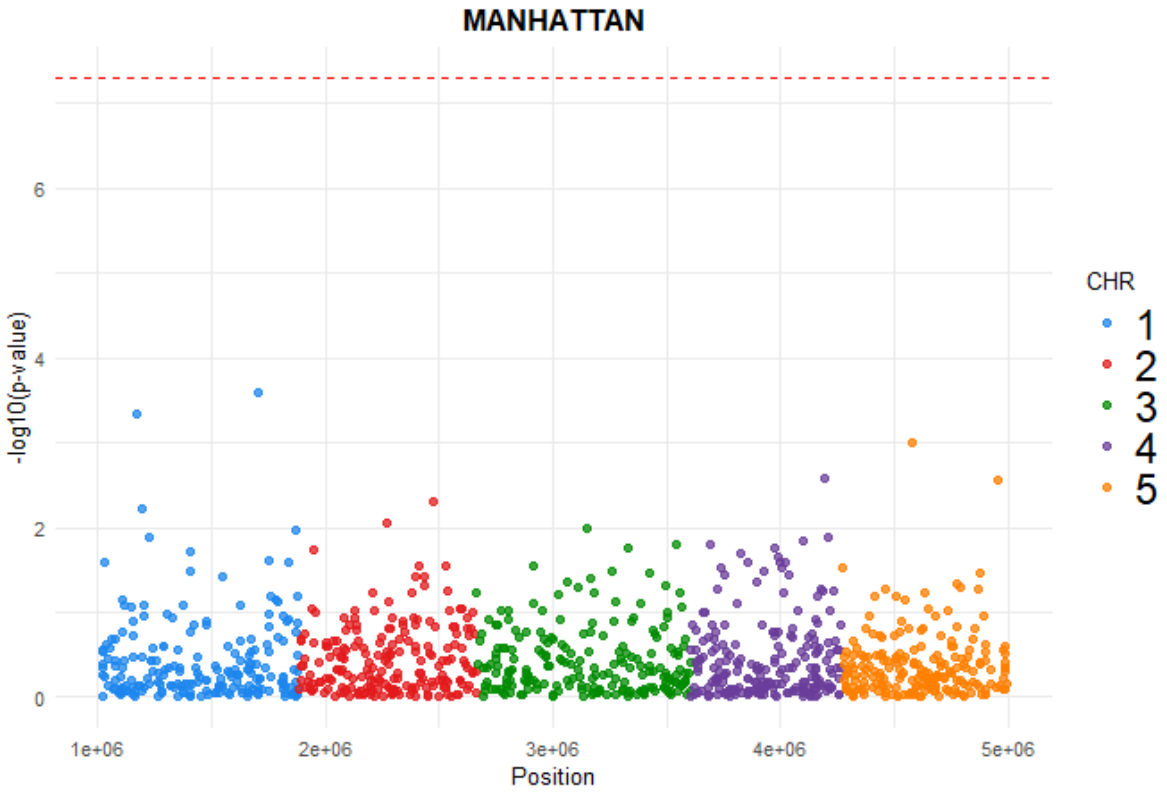


5.1.5 Manhattan plot with user-specified theme specifications

Command:

```
flex_manhattan(example_data2,
           user_legend_title = "CHR",
           user_title = "MANHATTAN",
           user_plot_theme_specs = theme(
             legend.text = element_text(size = 20),
             plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, face = "bold"
             )))
```

Output:

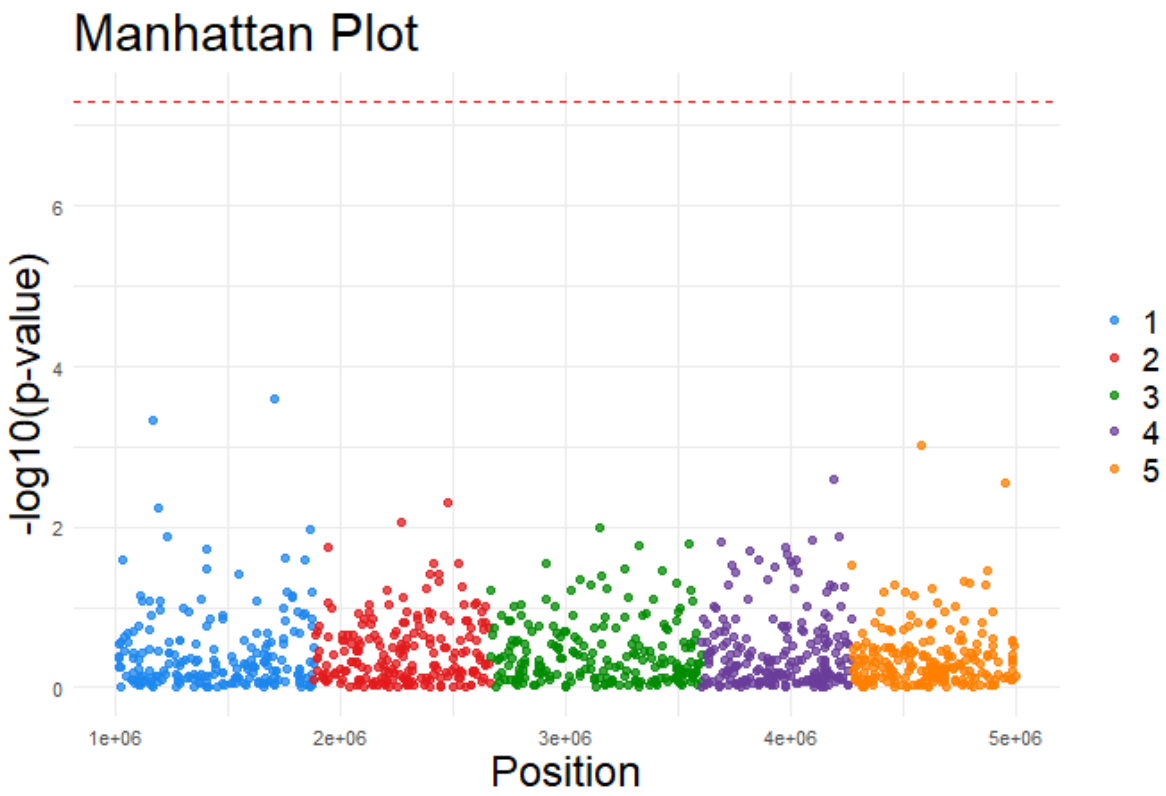


5.1.6 Manhattan plot with the title of the legend removed

Command:

```
flex_manhattan(example_data2,
           user_legend_title = NULL,
           user_plot_theme_specs = theme(
             title = element_text(size = 20),
             legend.text = element_text(size = 15)
           ))
```

Output:

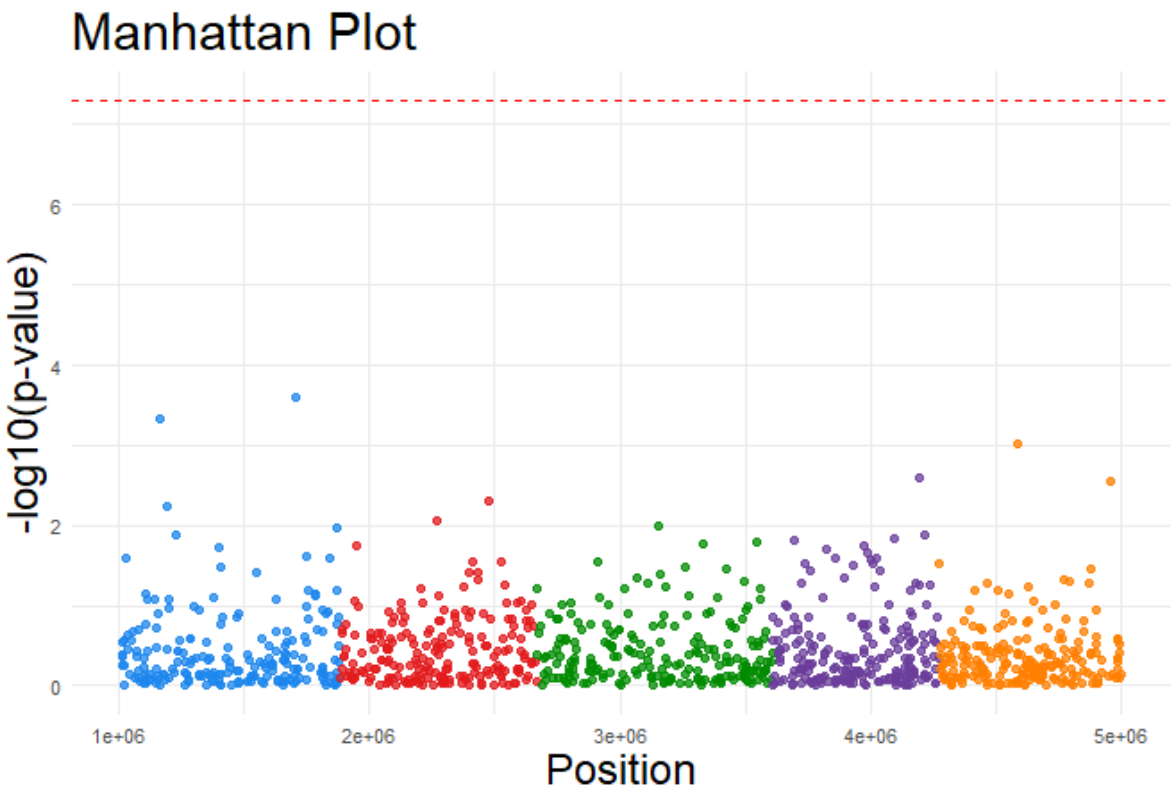


5.1.7 Manhattan plot with the legend removed

Command:

```
flex_manhattan(example_data2,
           user_plot_theme_specs = theme(
             title = element_text(size = 20),
             legend.text = element_text(size = 20),
             legend.title = element_text(size = 15),
             legend.position = "none"
            ))
```

Output:

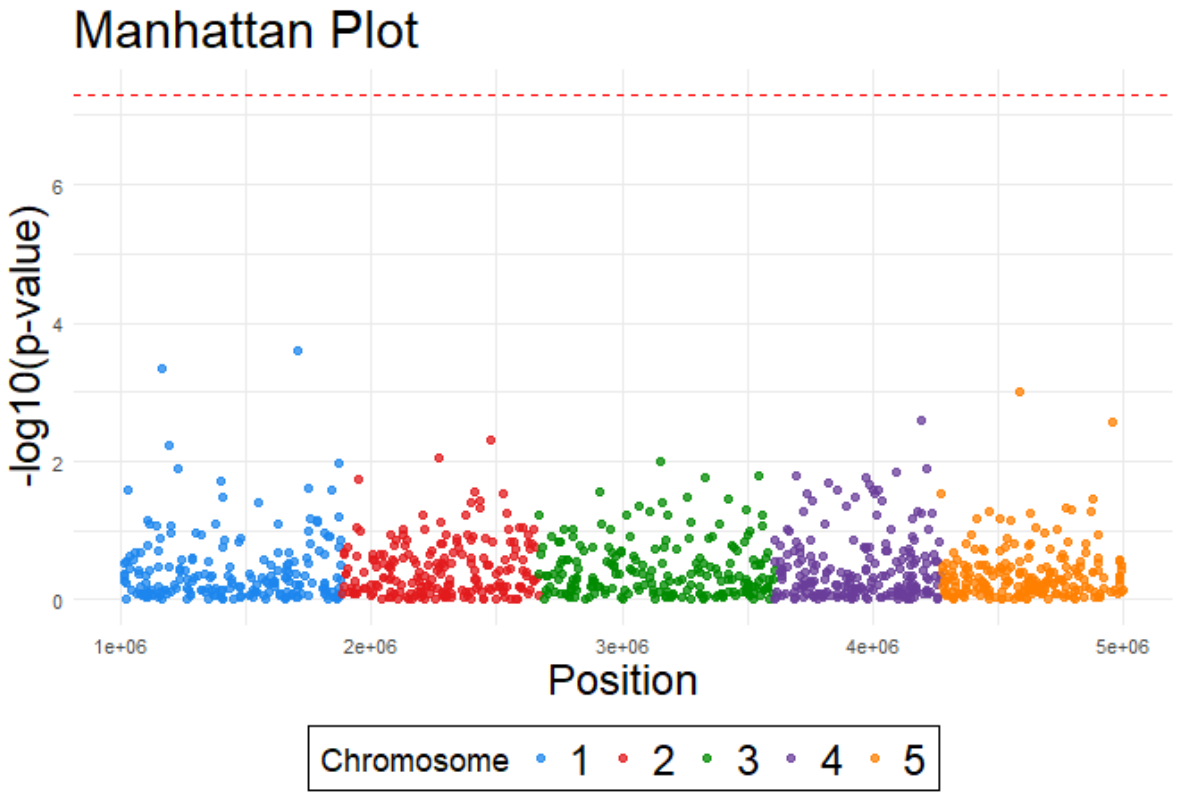


5.1.8 Manhattan plot with the formatted legend

Command:

```
flex_manhattan(example_data2,
               interactive = FALSE,
               user_plot_theme_specs = theme(
                 title = element_text(size = 20),
                 legend.text = element_text(size = 20),
                 legend.title = element_text(size = 15),
                 legend.position = "bottom",
                 legend.box.background = element_rect(colour = "black")
               ))
```

Output:

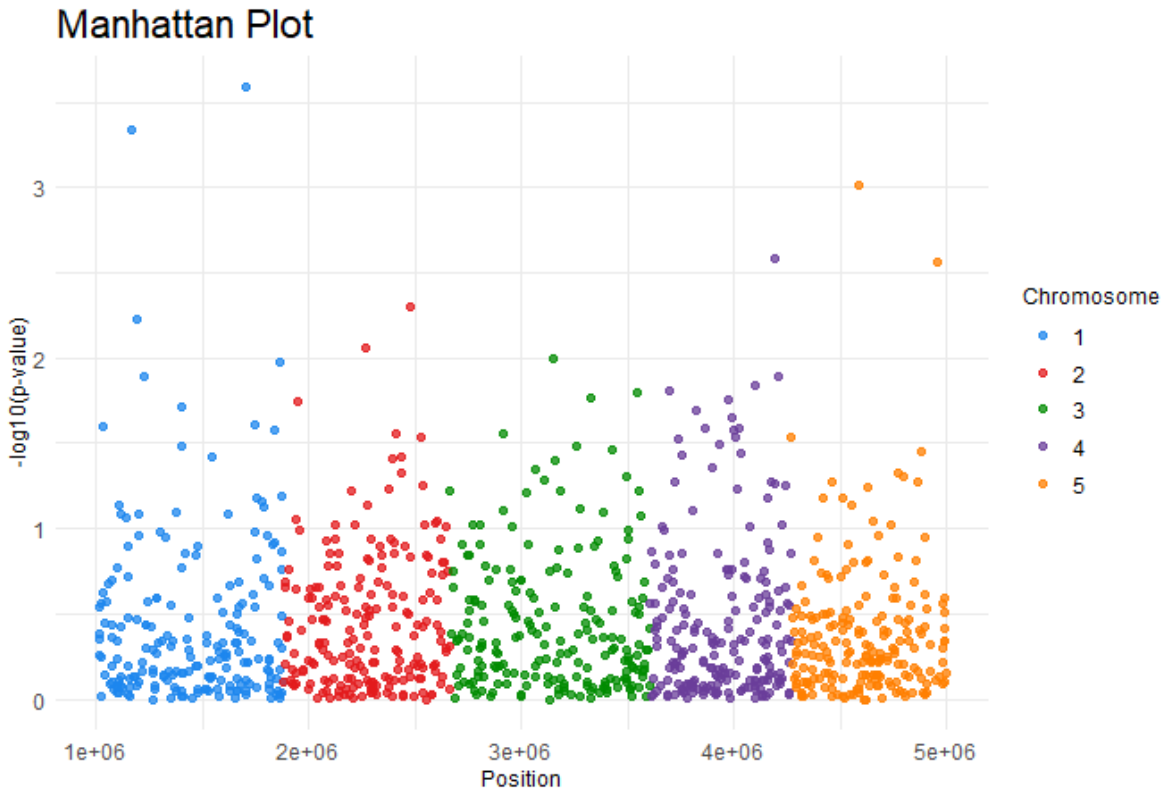


5.1.9 Manhattan plot without threshold line

Command:

```
flex_manhattan(example_data2, user_y_cutoff = NULL)
```

Output:

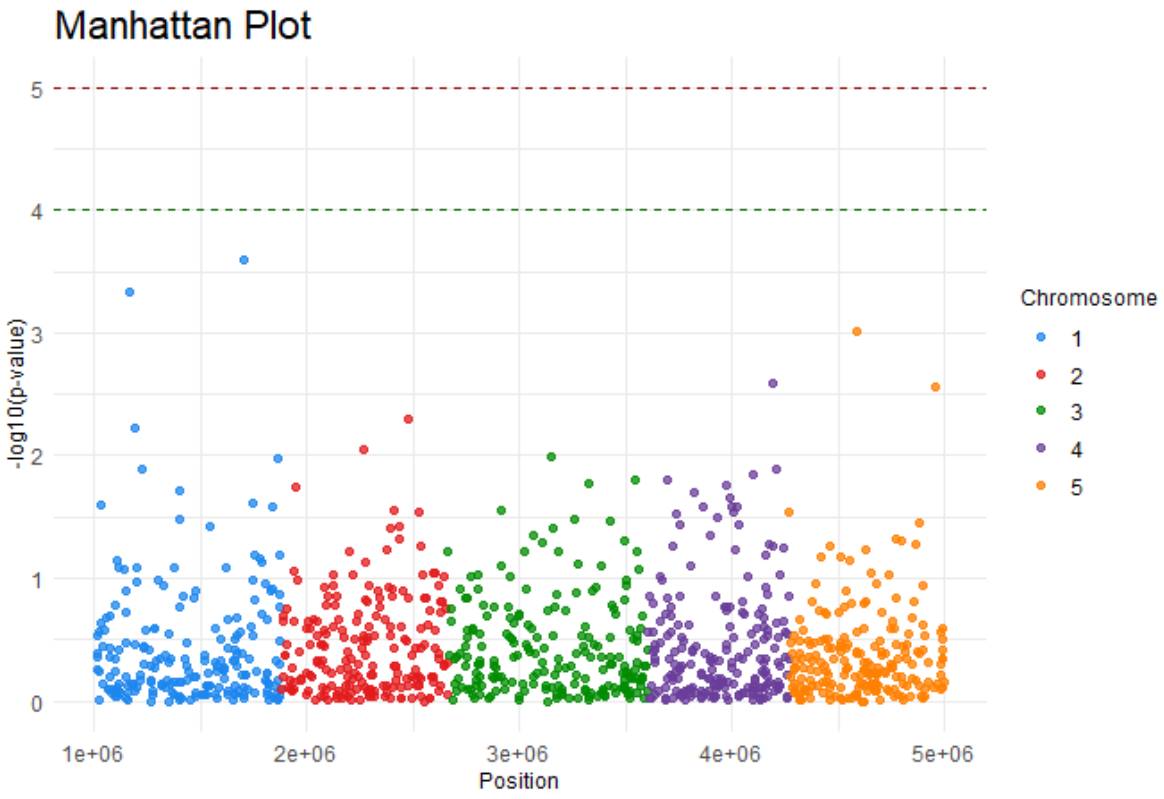


5.1.10 Manhattan plot with multiple threshold lines

Command:

```
flex_manhattan(example_data2,
           user_y_cutoff = c(5, 4),
           user_y_cutoff_color = c("darkred", "darkgreen"))
```

Output:



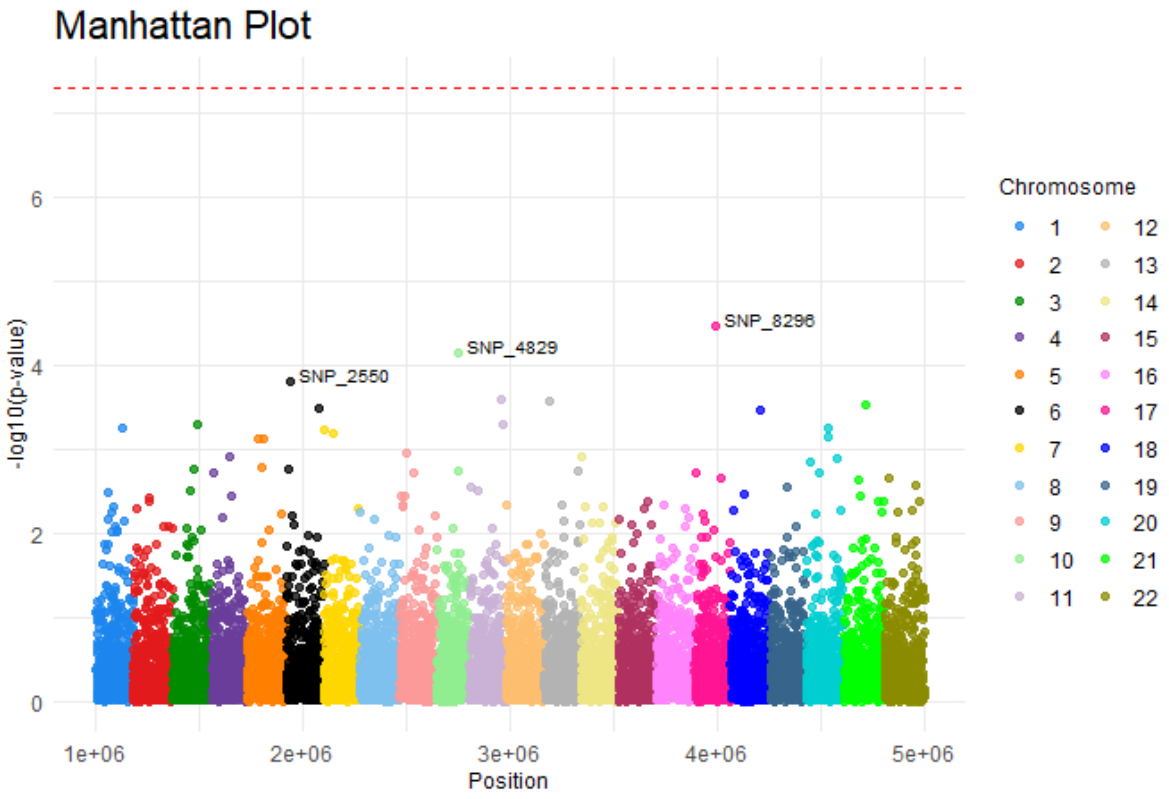
5.1.11 Manhattan plot with user-specified SNP annotations

Command:

```
annotated_snps <- c("SNP_2550", "SNP_4829", "SNP_8296")

flex_manhattan(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE)
```

Output:



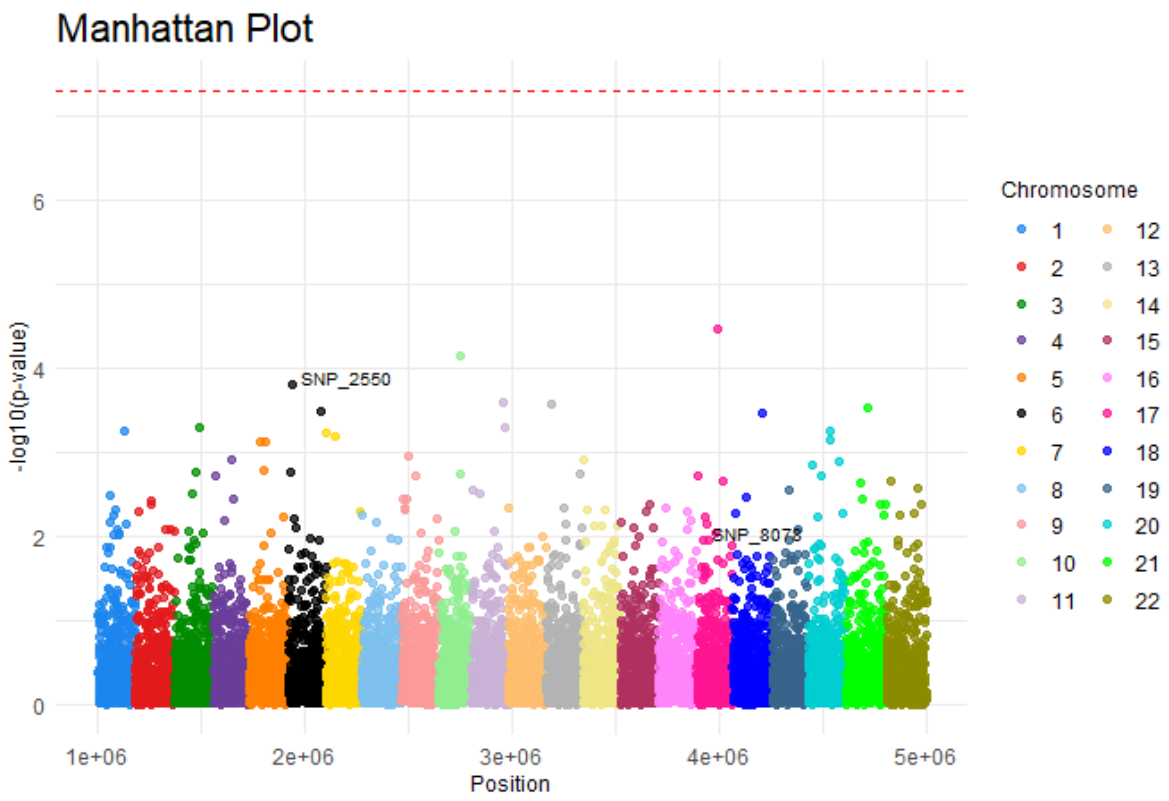
5.1.12 Manhattan plot with user-specified SNP annotations, given locations

Command:

```
annotated_pos <- c("1939279", "3919913")

flex_manhattan(example_data1, annotate_data = annotated_pos, annotate_labels = TRUE,
annotate_column = "POS")
```

Output:



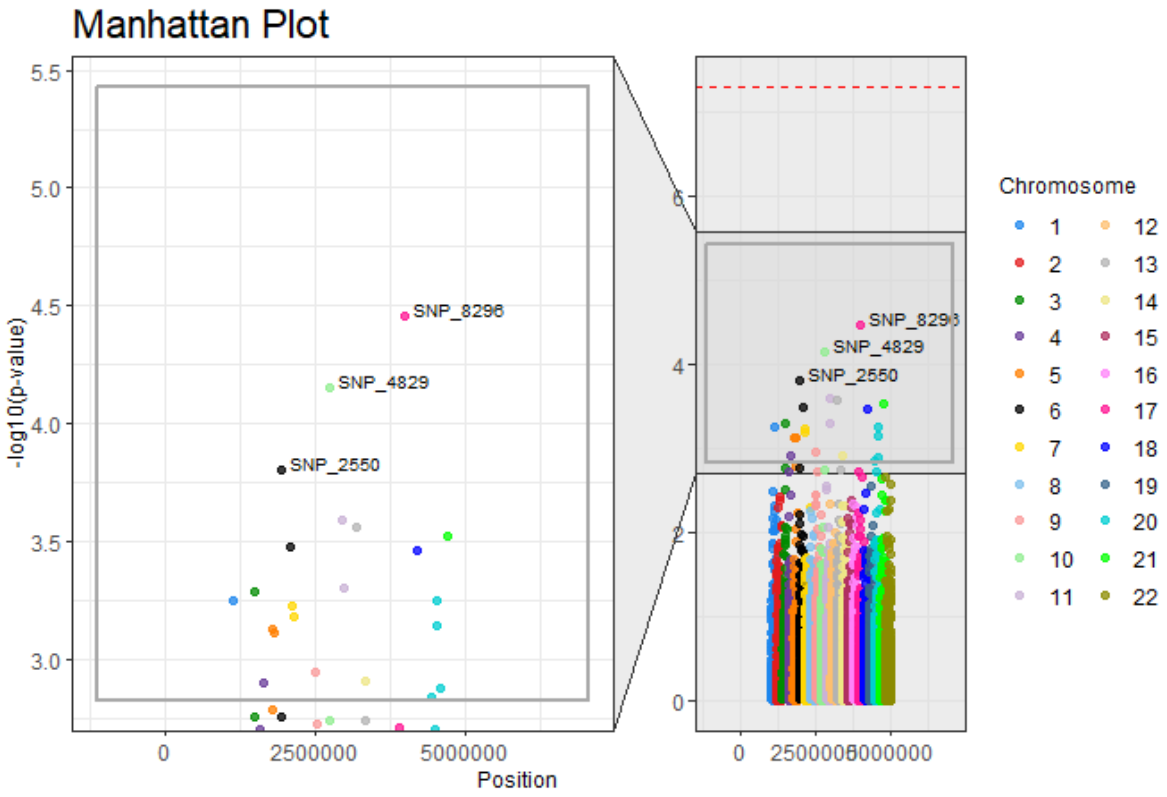
5.1.13 Manhattan plot with user-specified SNP annotations and zoom-in

Command:

```
annotated_snps <- c("SNP_2550", "SNP_4829", "SNP_8296")

flex_manhattan(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE,
zoom_on_annotations = TRUE, zoom_margin = 1.5,
user_plot_theme = theme_bw())
```

Output:



5.2 flex_qqplot()

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 (KS) test result annotation.

5.2.1 Arguments

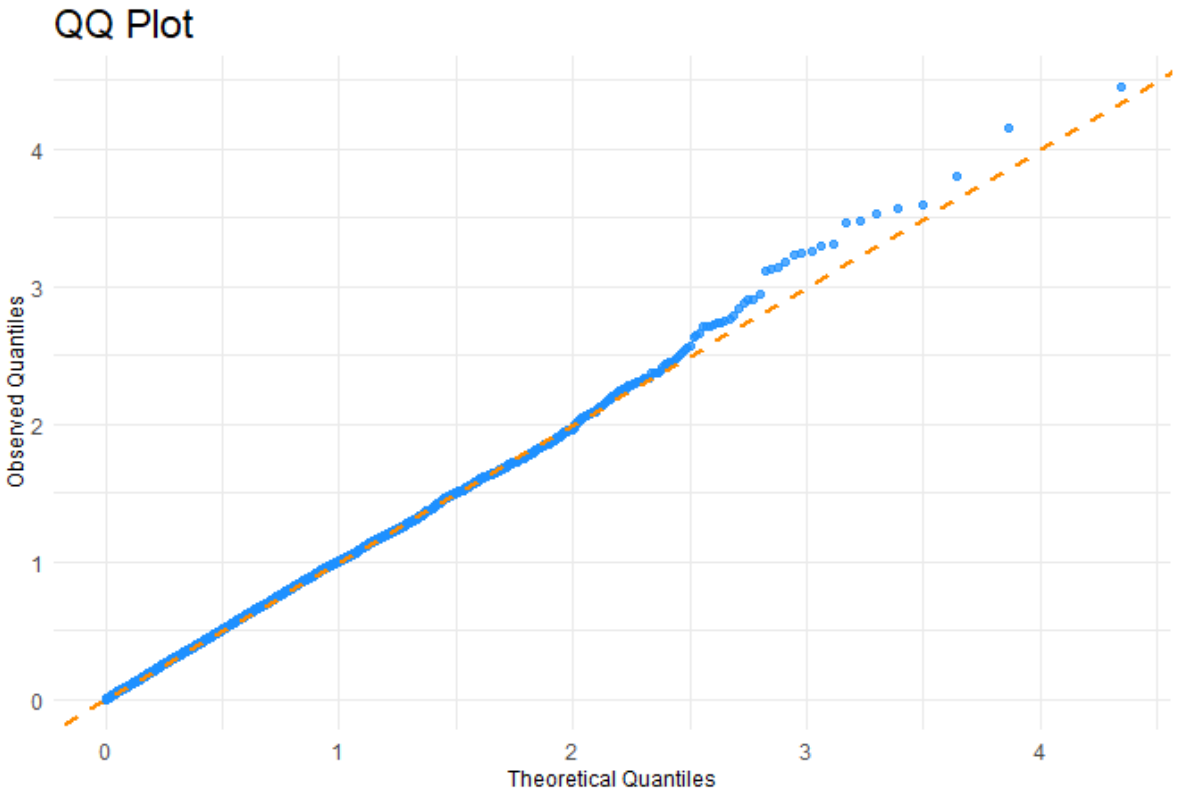
- gwas_data: A data frame containing columns for SNP ID ('SNP'), chromosome ('CHR'), position ('POS'), and p-values ('P_VALUE').
- interactive: Logical, whether to return an interactive plot (TRUE) or static ggplot (FALSE).
- display_ks: Logical, whether to display the Kolmogorov-Smirnov (KS) test result.
- 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.

5.2.2 Default qq plot

Command:

```
flex_qqplot(example_data1)
```

Output:

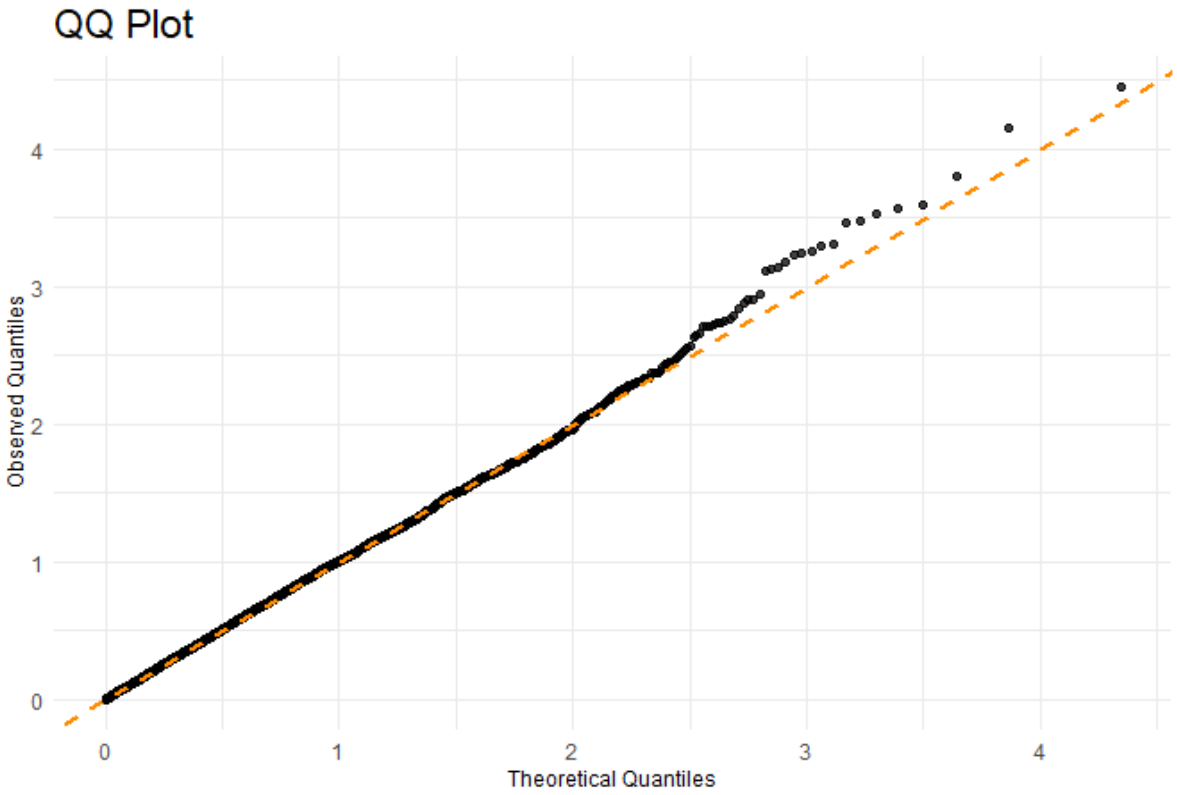


5.2.3 qq plot with use-specified colors

Command:

```
flex_qqplot(example_data1, user_colors = c("black", "darkorange"))
```

Output:

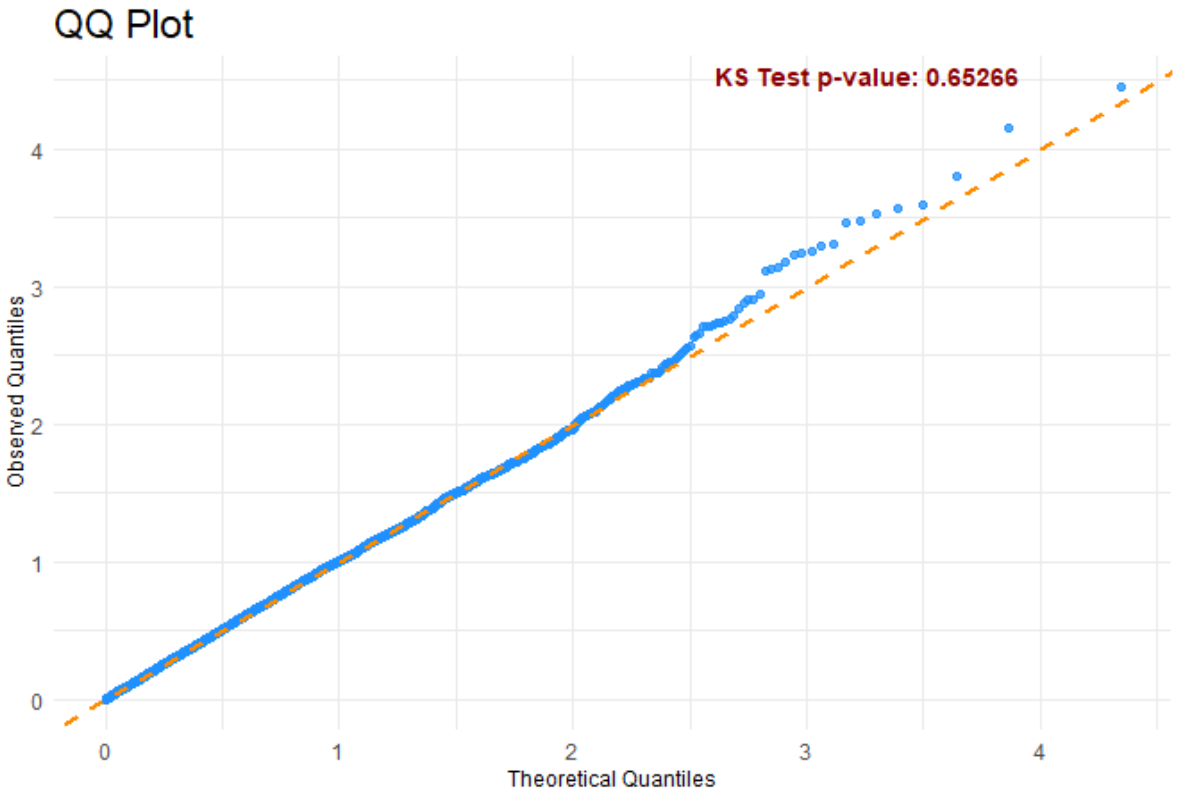


5.2.4 qq plot with KS test results

Command:

```
flex_qqplot(example_data1, display_ks = TRUE, interactive = FALSE)
```

Output:



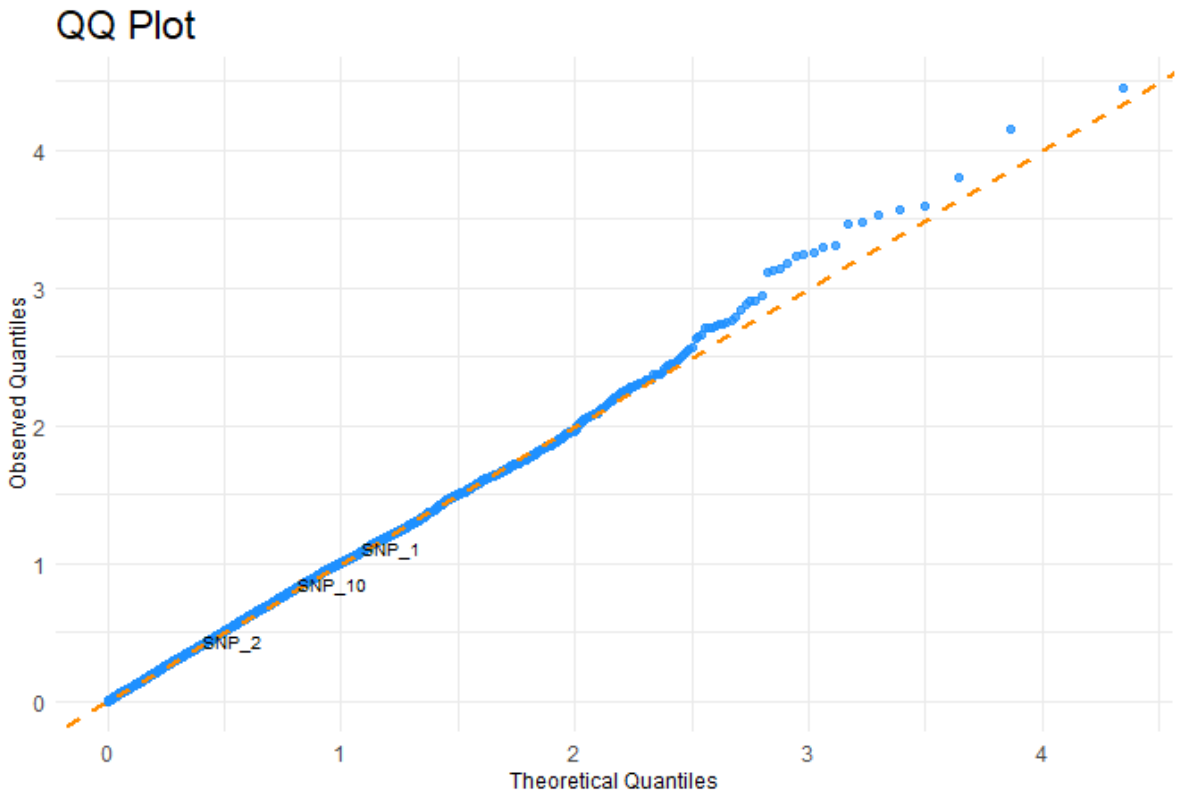
5.2.5 qq plot with annotated user-specified SNPs

Command:

```
annotated_snps <- c("SNP_1", "SNP_2", "SNP_10")

flex_qqplot(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE)
```

Output:



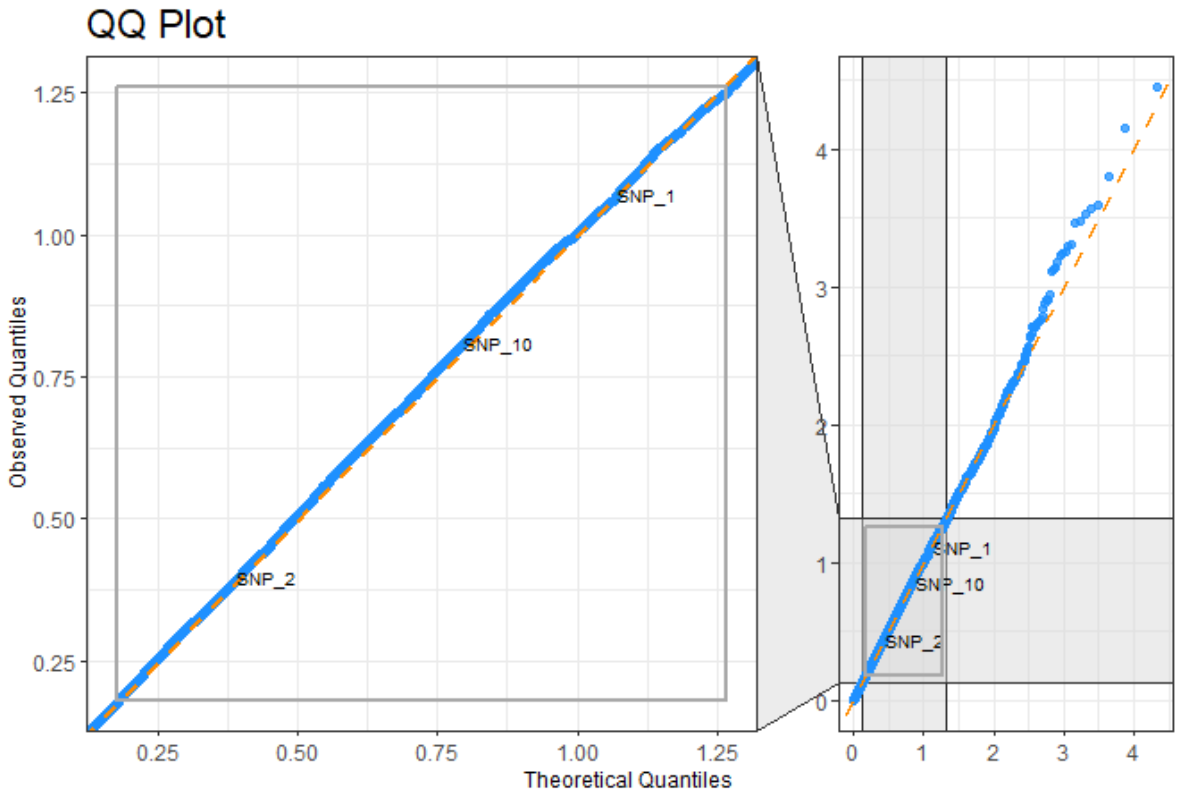
5.2.6 qq plot with user-specified SNP annotations, theme and zoom-in

Command:

```
annotated_snps <- c("SNP_1", "SNP_2", "SNP_10")

flex_qqplot(example_data1, annotate_data = annotated_snps, annotate_labels = TRUE,
zoom_on_annotations = TRUE, zoom_margin = 0.3, user_plot_theme = theme_bw())
```

Output:



5.3 flex_accuracy()

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

5.3.1 Arguments

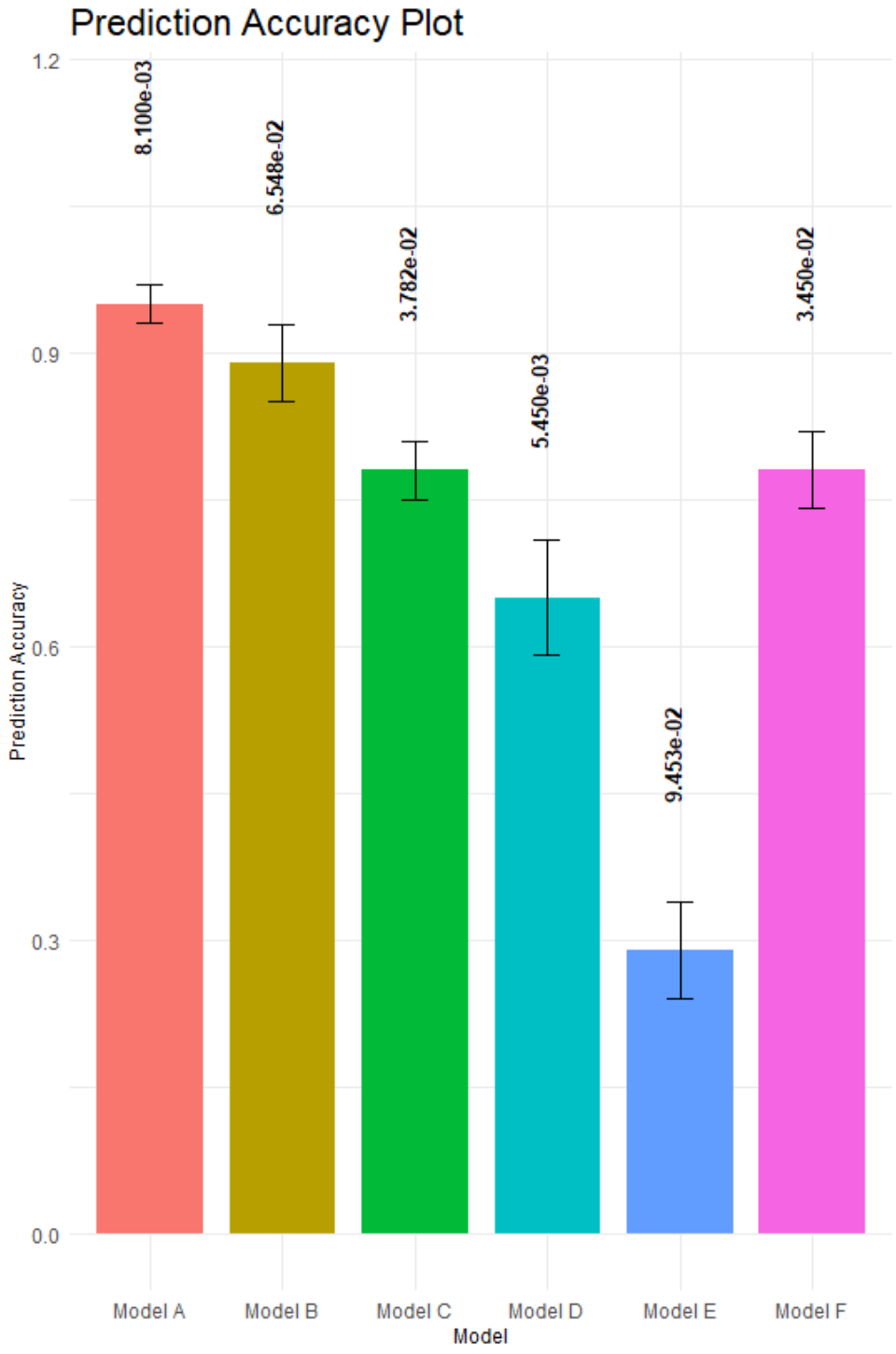
- accuracy_data: A data frame containing three columns: 'model' (factor or character vector of model names), 'rsquared' (numeric vector of prediction accuracies), and 'p_value' (numeric vector of p-values).
- conf_level: A numeric value between 0 and 1 to indicate level of confidence; default is 0.95.
- 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.

5.3.2 Default accuracy plot

Command:

```
flex_accuracy(example_data3)
```

Output:

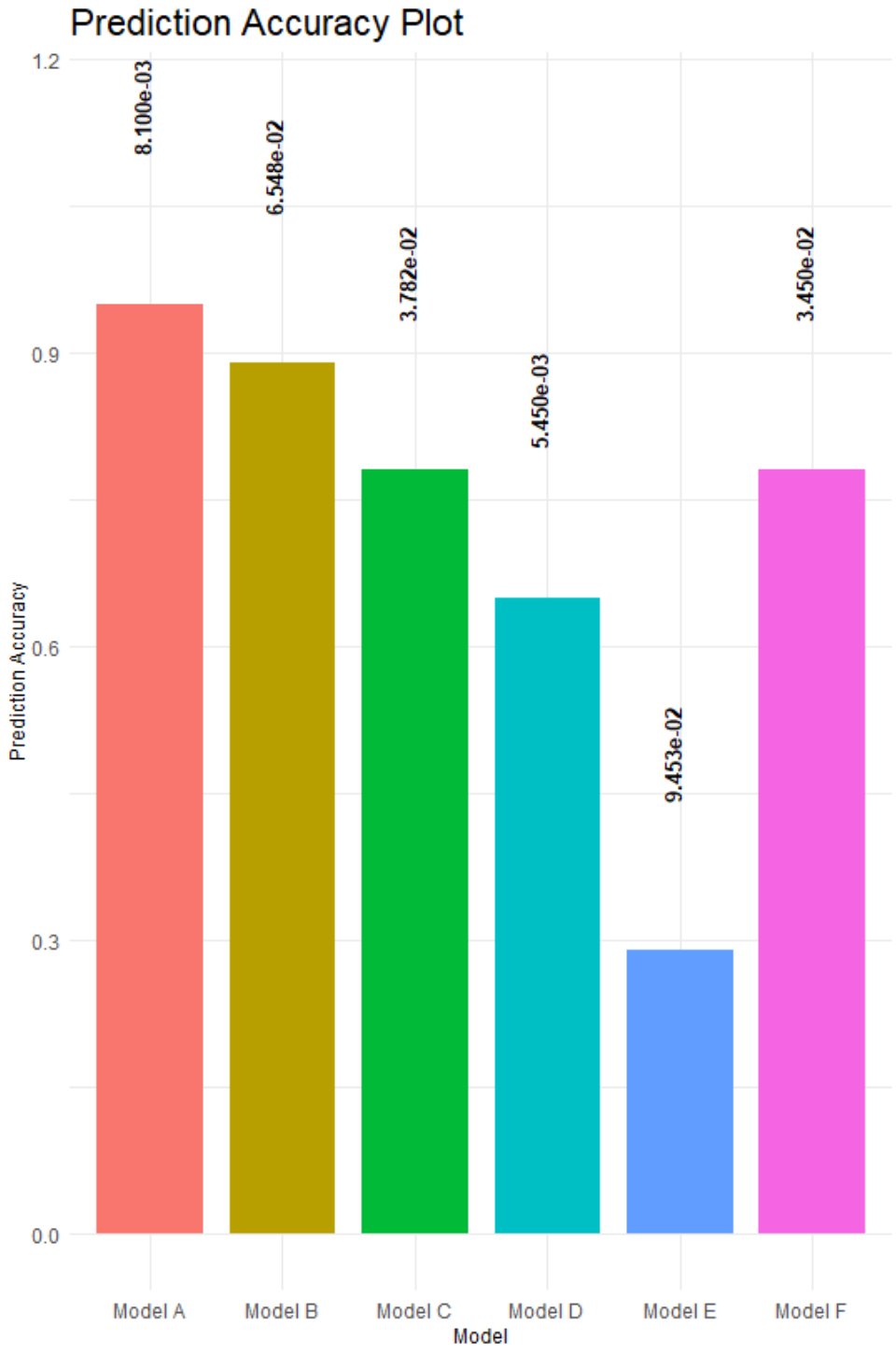


5.3.3 Accuracy plot without displaying a CI

Command:

```
flex_accuracy(example_data3, display_CI = FALSE)
```

Output:

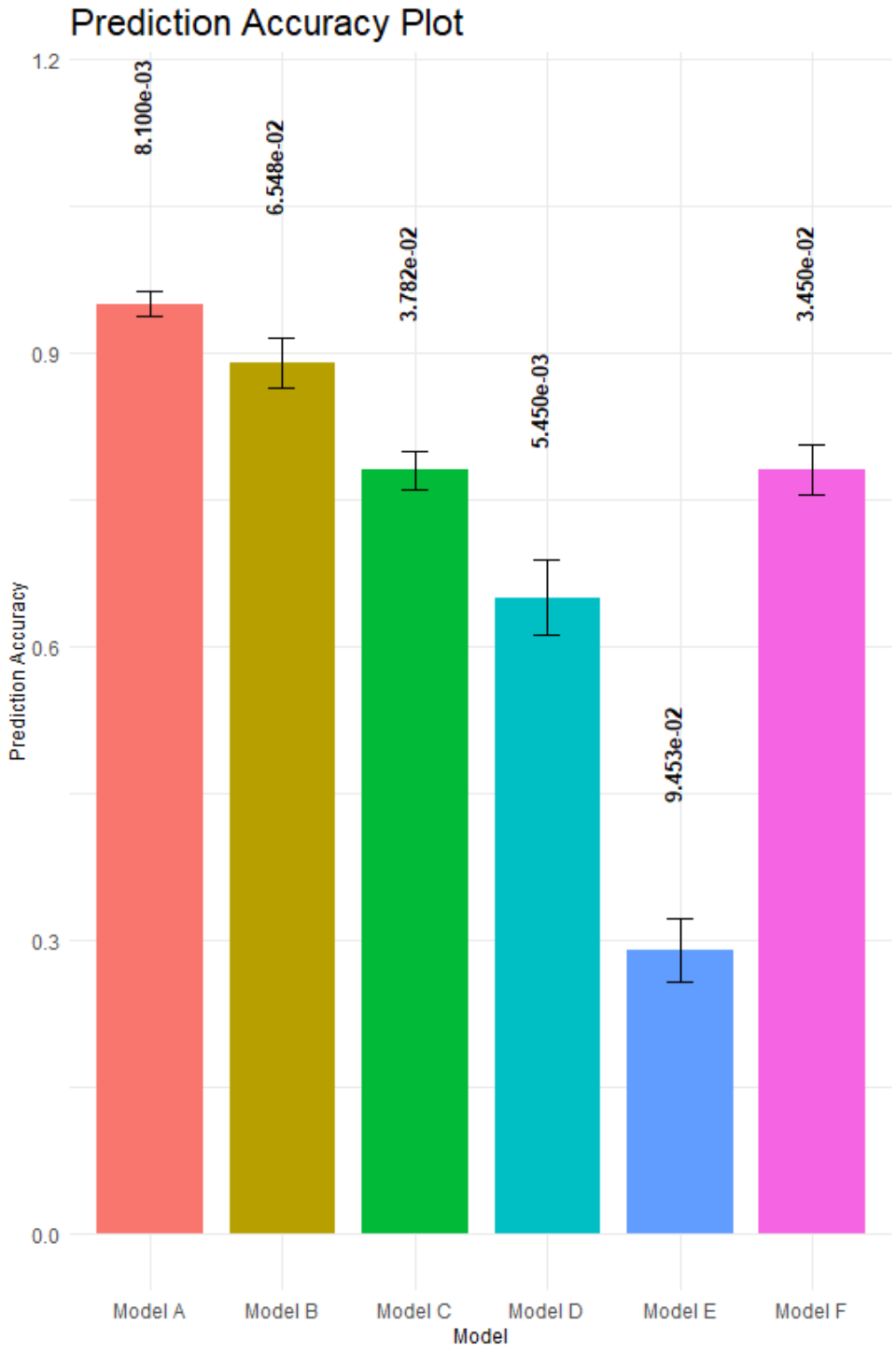


5.3.4 Accuracy plot with user-specified confidence level

Command:

```
flex_accuracy(example_data3, conf_level = 0.8)
```

Output:

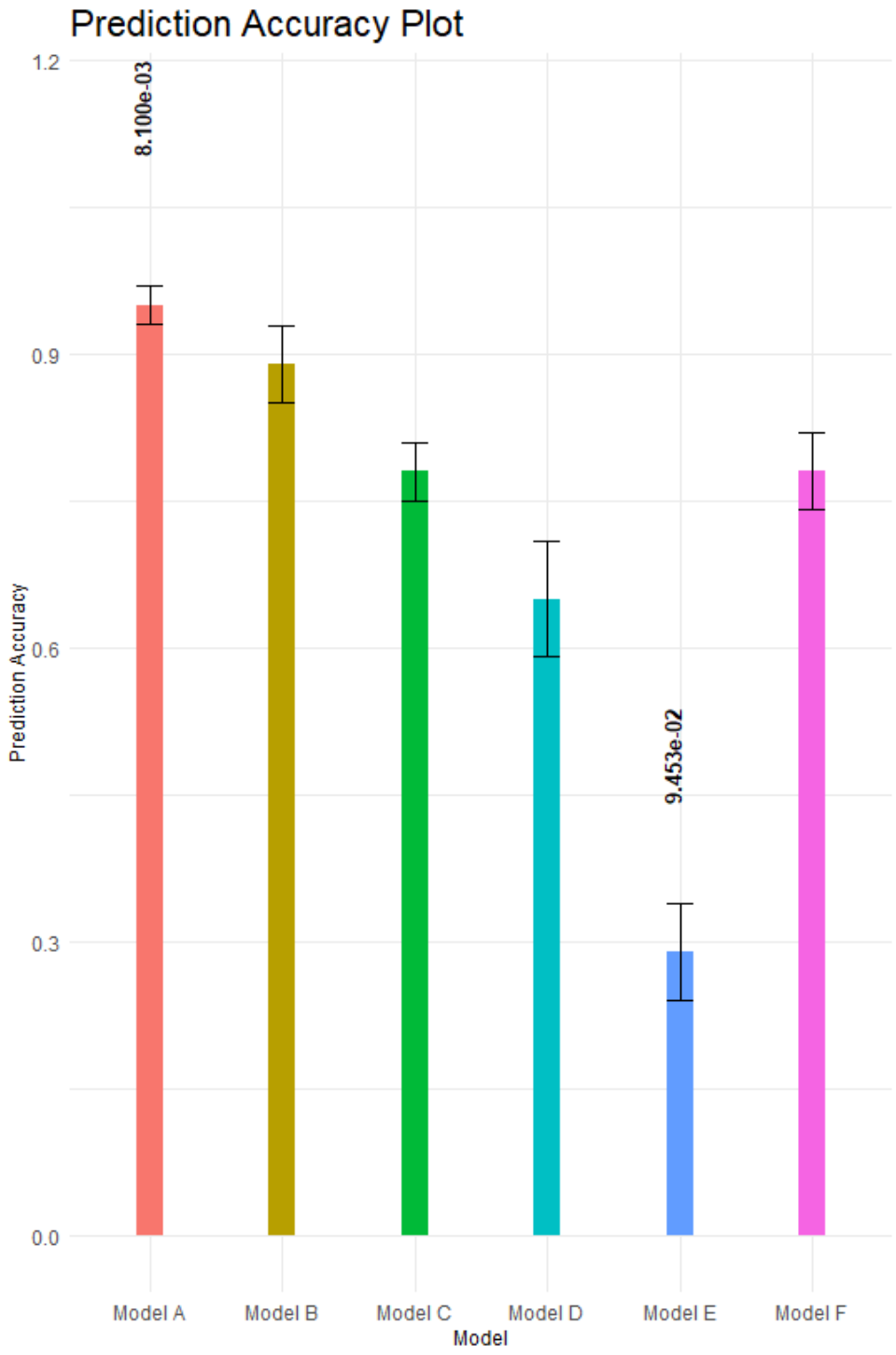


5.3.5 Accuracy plot showing only user-specified models' p-values with thinner bars

Command:

```
flex_accuracy(example_data3, display_p_values = TRUE,
              Models_to_display_p_values = c("Model A", "Model E"),
              user_bar_width = 0.2)
```

Output:

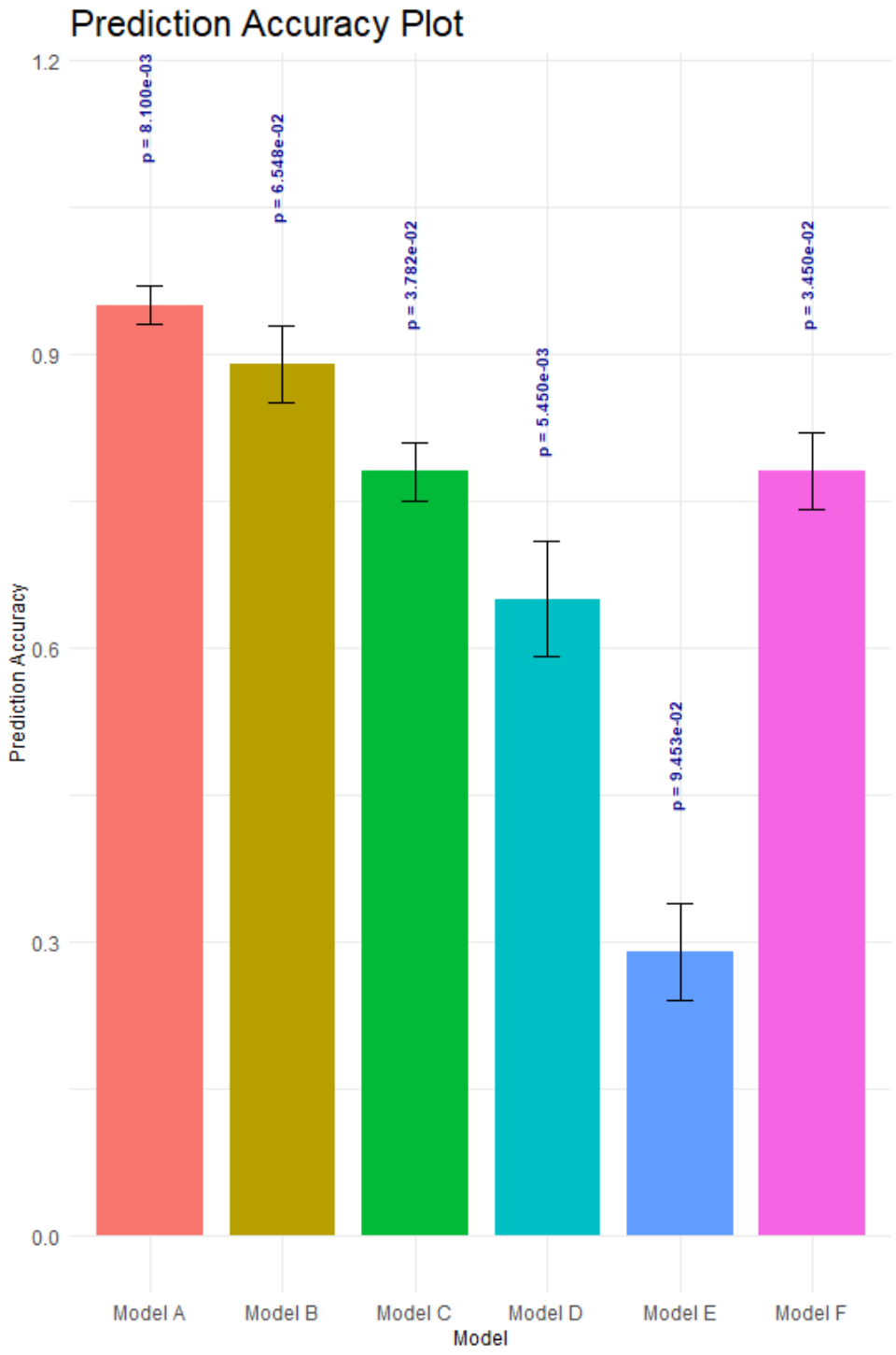


5.3.6 Accuracy plot with optionally added p-value annotations

Command:

```
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))
```

Output:

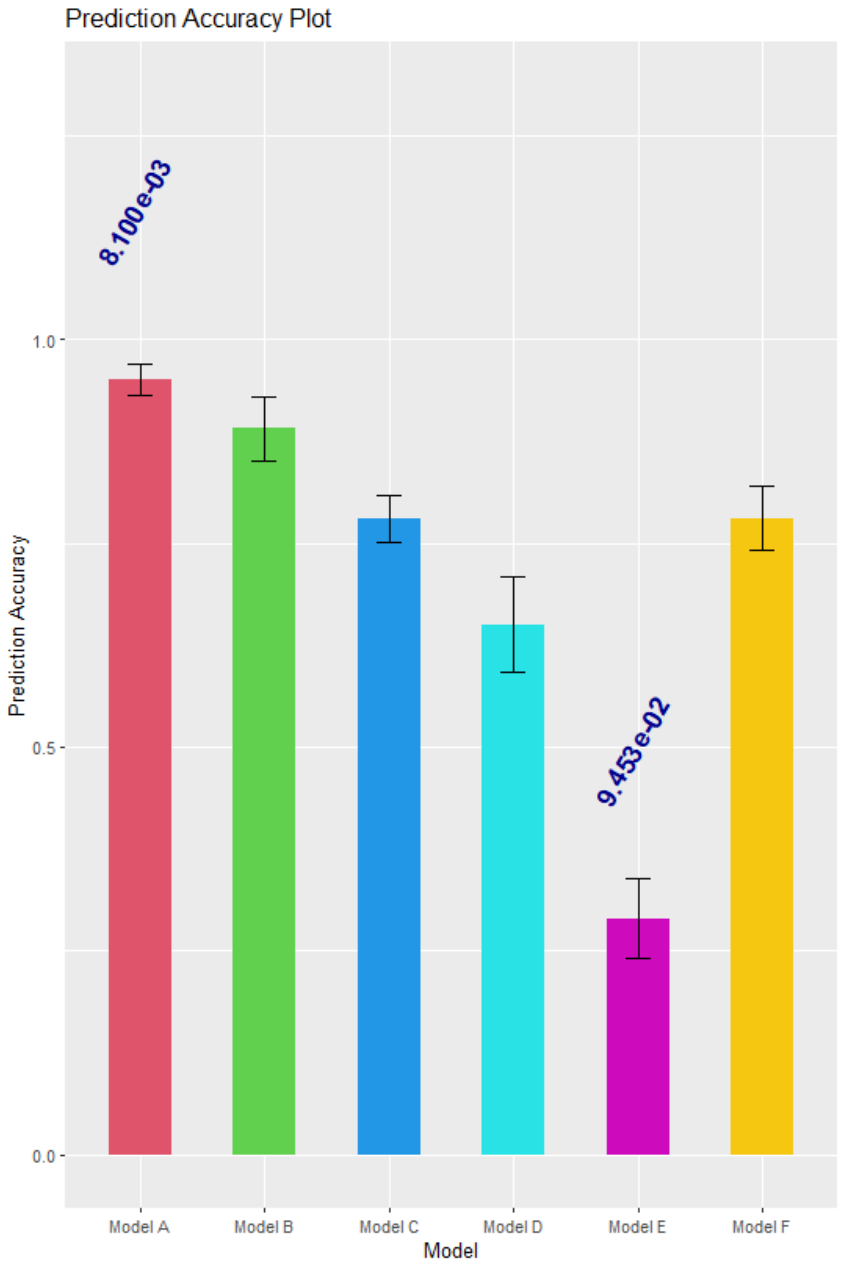


5.3.7 Accuracy plot with additional plotting features

Command:

```
flex_accuracy(example_data3, user_colors = c(2,3,4,5,6,7),
              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(scale_y_continuous(limits = c(0, 1.3)),
                                            theme_grey()))
```

Output:

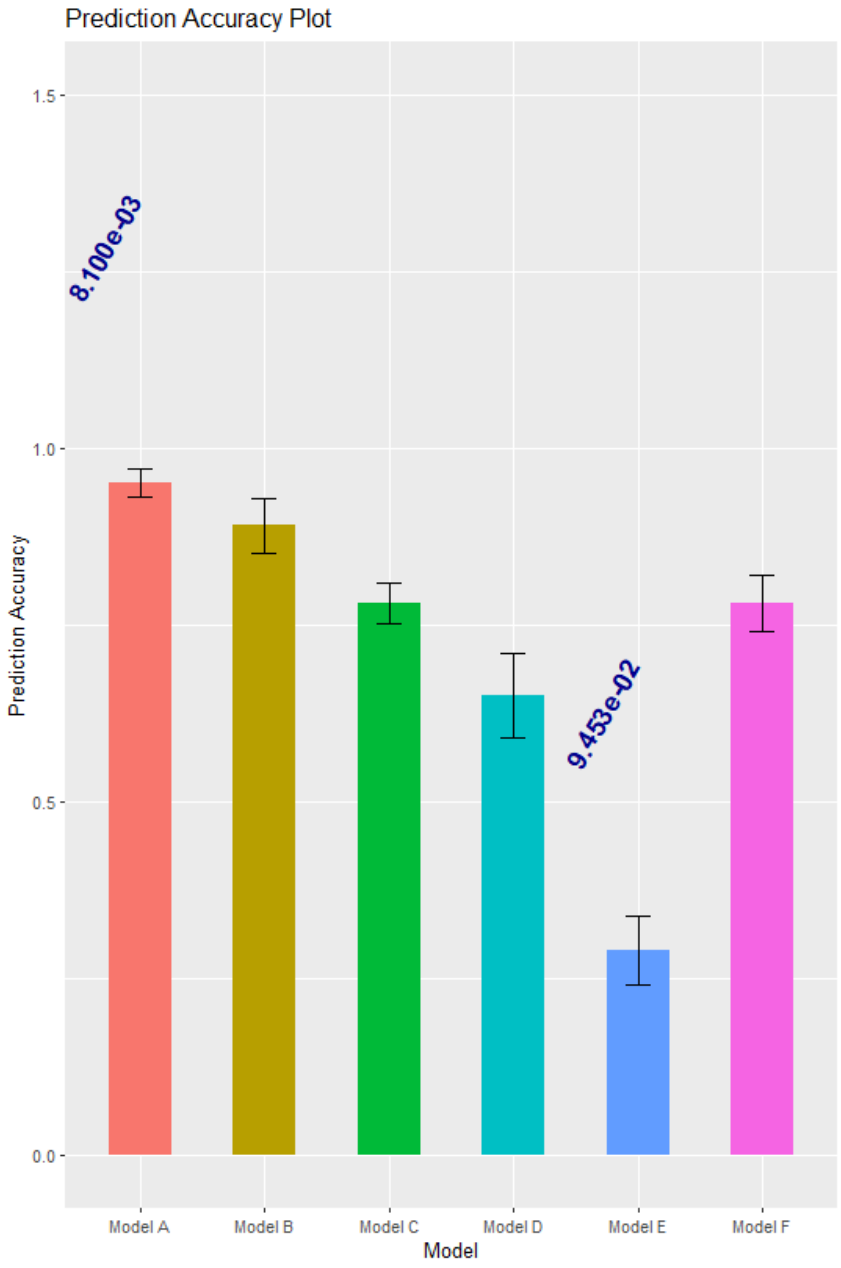


5.3.8 Accuracy plot with nudged p value annotations

Command:

```
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(scale_y_continuous(limits = c(0, 1.5)),
                                            theme_grey()))
```

Output:



5.4 flex_accuracy_diff()

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.

5.4.1 Arguments

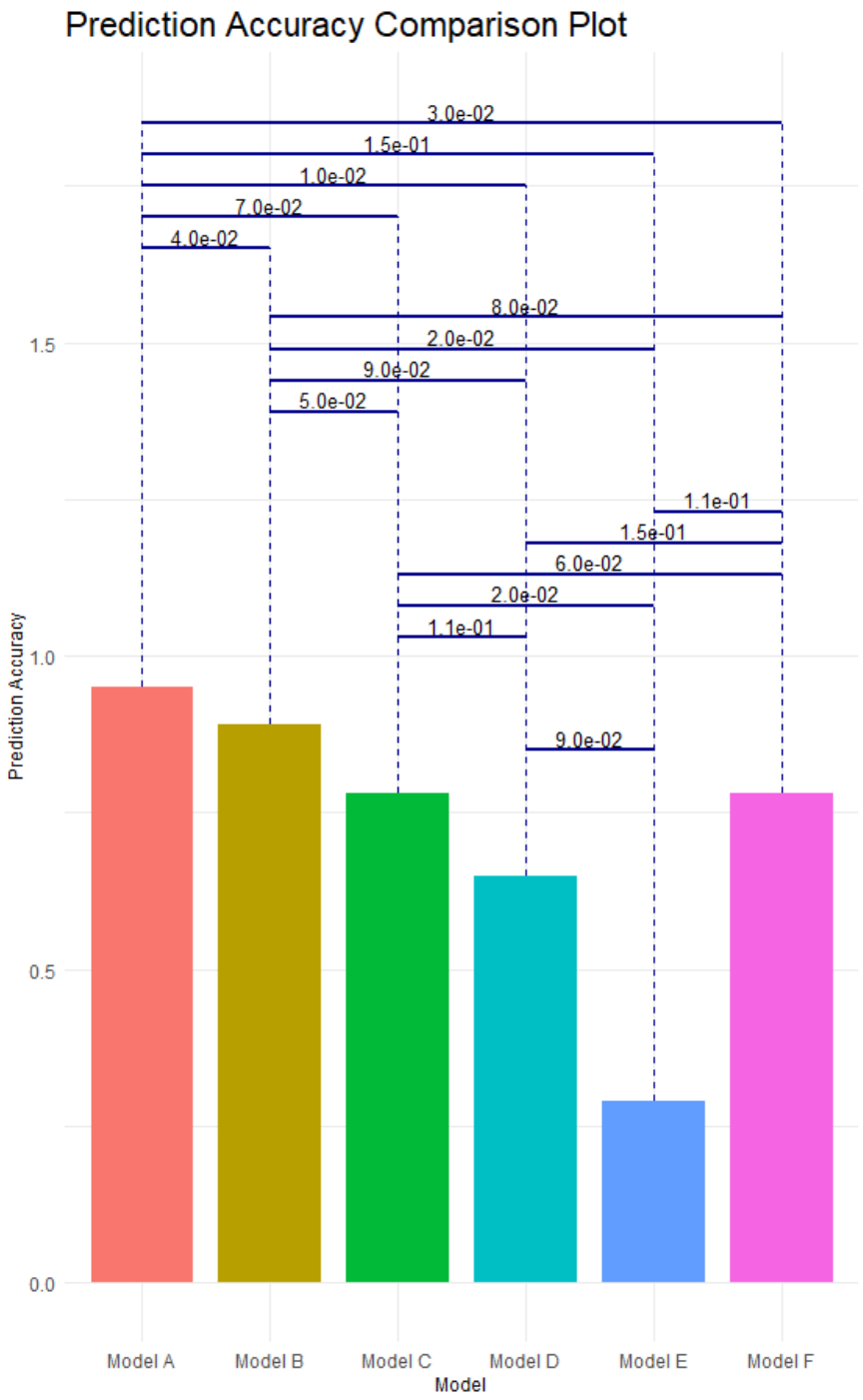
- accuracy_data: A data frame containing the models' R-squared values with two columns.'Model': Factor or character vector specifying model names.'R_squared': Numeric vector specifying R-squared values for each model.
- comparison_data: A data frame containing pairwise comparisons with three columns.'Compare1' and 'Compare2': Character or factor columns specifying the pair of models being compared. 'p_value': Numeric vector specifying p-values for the comparisons.
- 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.

5.4.2 Default accuracy comparison plot

Command:

```
flex_accuracy_diff(accuracy_data = example_data4, comparison_data = example_data5)
```

Output:

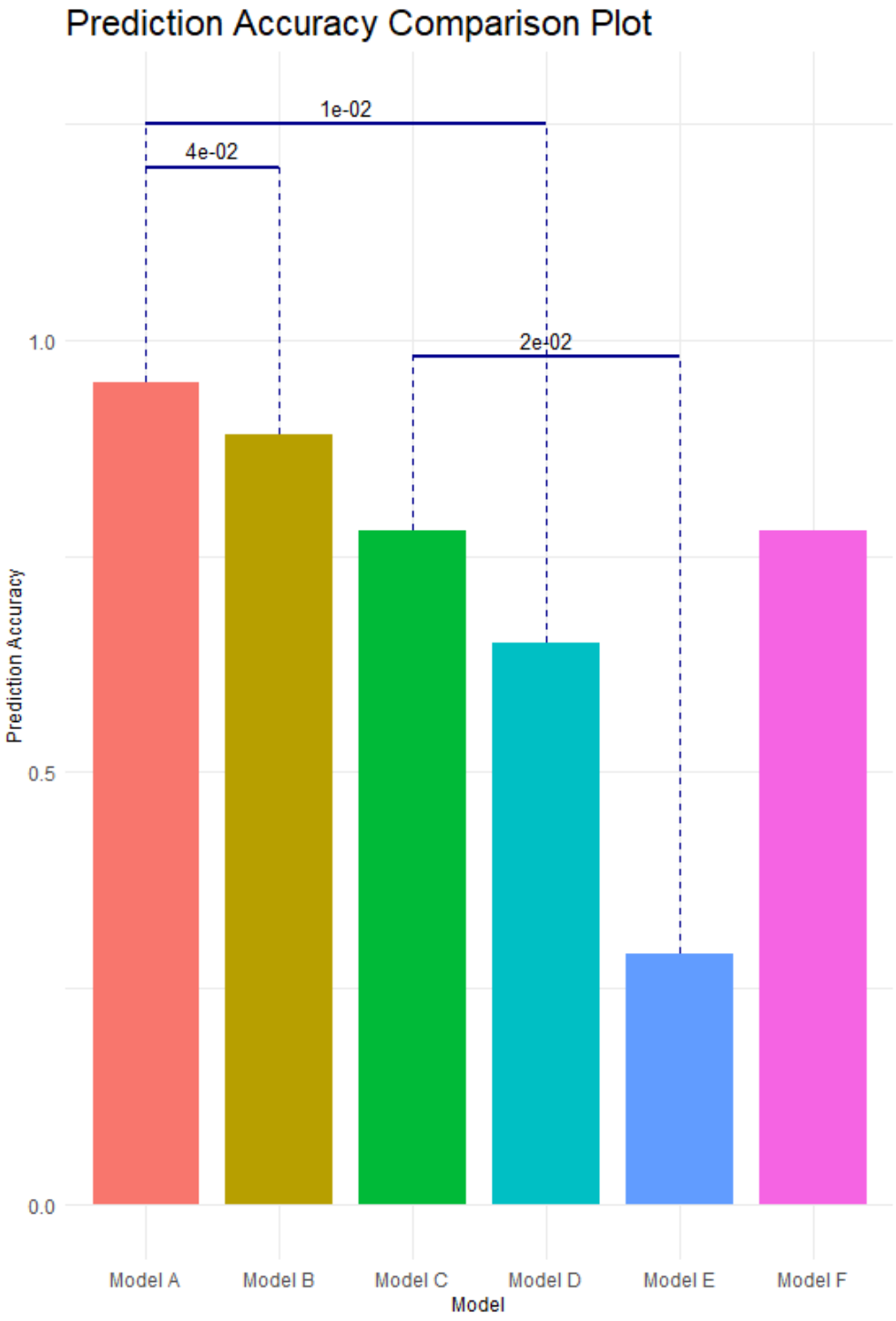


5.4.3 Accuracy comparison plot showing only user-specified models' p-values

Command:

```
flex_accuracy_diff(accuracy_data = example_data4, comparison_data = example_data5,
            models_to_display_p_values = c("Model A Model B", "Model E Model C",
            "Model D Model A"))
```

Output:

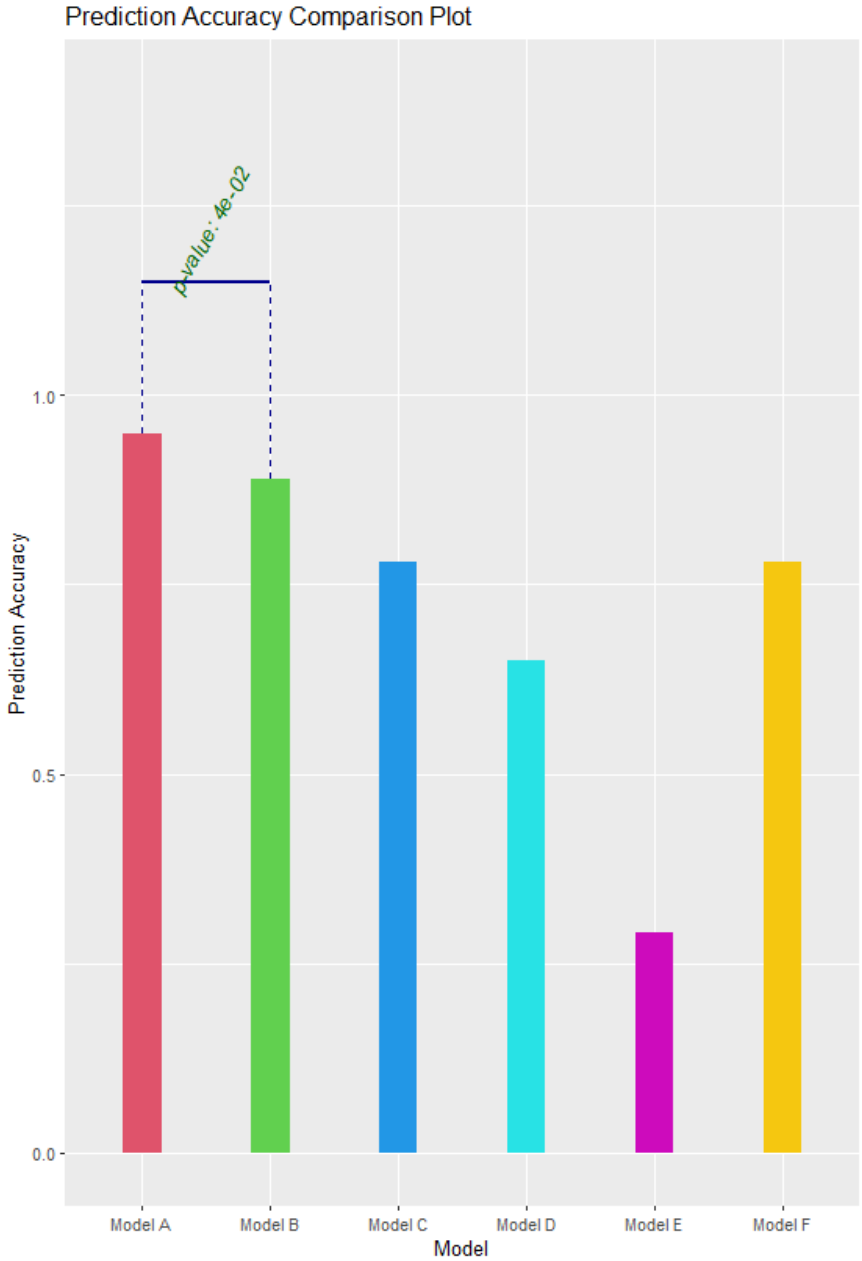


5.4.4 Accuracy comparison plot with user-specified, additional plotting features

Command:

```
flex_accuracy_diff(accuracy_data = example_data4, comparison_data = example_data5,
                   models_to_display_p_values = c("Model A Model B"),
                   user_colors = c(2,3,4,5,6,7),
                   user_bar_width = 0.3,
                   additional_ggplot_args = list(scale_y_continuous(limits = c(0, 1.4)),
                                                 theme_grey()),
                   user_p_value_prefix = "p-value: ",
                   geom_text_args = list(color = "darkgreen",
                                         fontface = "italic", size = 4, angle = 60,
                                         nudge_y = 0.05))
```

Output:



5.5 flex_accuracy_diff2()

This function generates a customizable bar plot to compare prediction accuracy (e.g. R-squared values) between multiple methods given traits. It displays confidence intervals for accuracy values (e.g. R squared) and their differences, and corresponding p-values specific to each trait of interest. This is a more detailed version of flex_accuracy_diff() output. Interactive plots can be created using Plotly.

5.5.1 Arguments

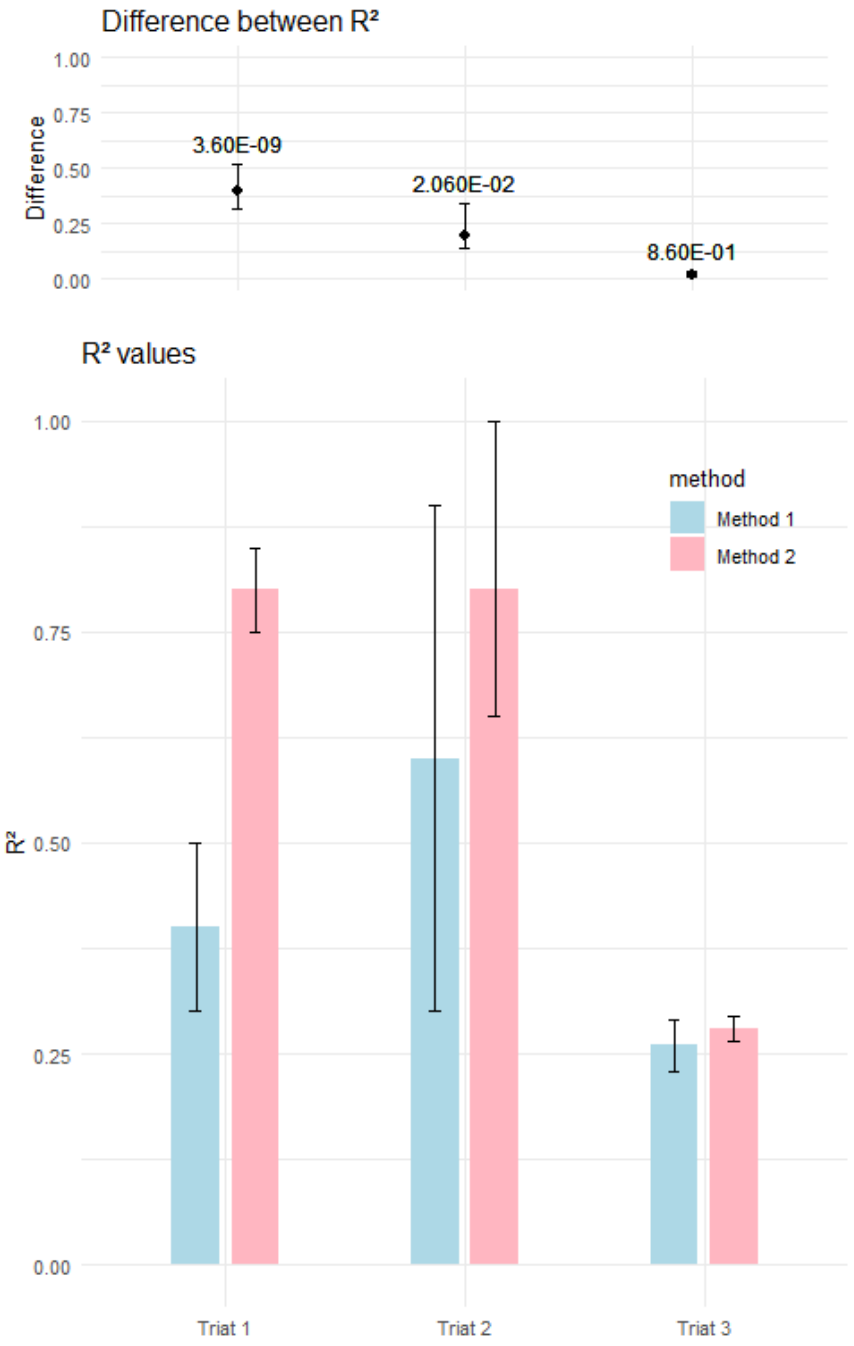
- accuracy_diff_data: A data frame containing nine columns and at least two rows: 'trait'(factor or character vector of trait names), 'method'(factor or character vector of method names (two methods)), 'R2'(numeric vector of accuracy of the methods for a trait), 'lower_limit_R2'(numeric vector of lower 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.

5.5.2 Default detailed accuracy difference plot

Command:

```
flex_accuracy_diff2(example_data6)
```

Output:

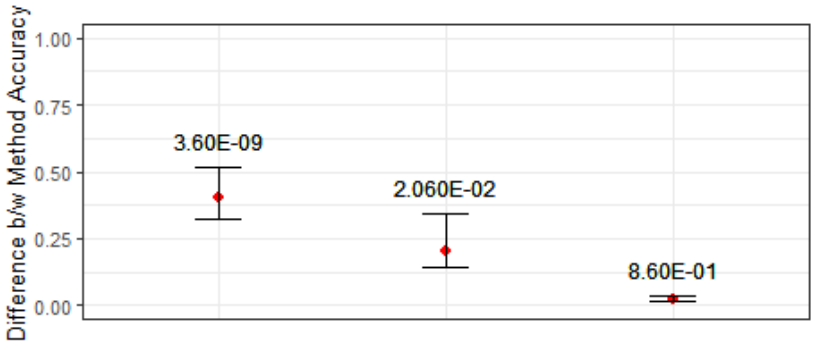


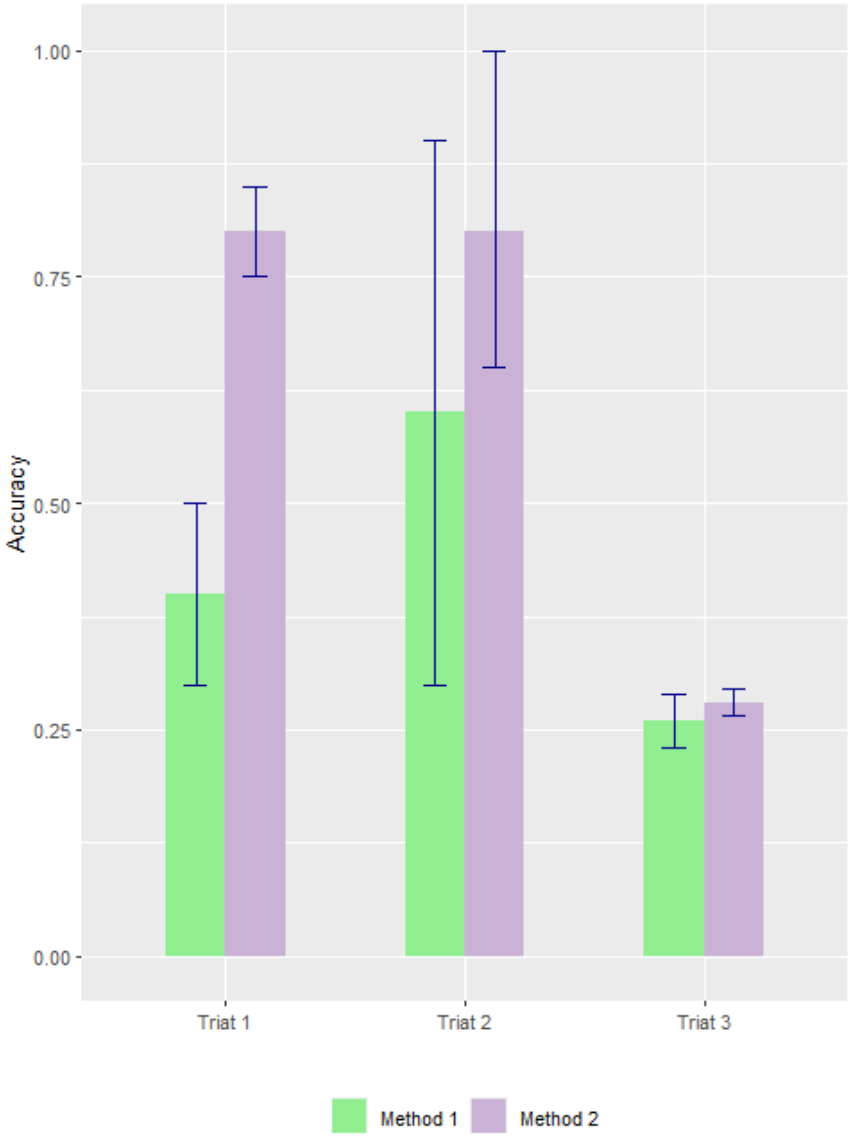
5.5.3 Detailed accuracy difference plot with user-specified colors, themes, legends, titles and other formatting

Command:

```
flex_accuracy_diff2(example_data6,
                user_colors = c("palegreen2", "#CAB2D6"),
                user_geom_bar = geom_bar(stat = "identity",
                                         position = position_dodge(0.5), width = 0.5),
                user_geom_point = geom_point(aes(x = trait, y = R2diff),
                                             color = "red", size = 2),
                user_accuracy_geom_errorbar = geom_errorbar(aes(ymin = lower_limit_R2,
                                             ymax = upper_limit_R2), color = "darkblue",
                                             width = 0.2, position = position_dodge(0.5)),
                user_accuracy_diff_geom_errorbar = geom_errorbar(width = 0.2,
                                                   color = "black"),
                user_accuracy_labs = labs(x = " ", y = "Accuracy", title = " "),
                user_accuracy_diff_labs = labs(x = " ",
                                             y = "Difference b/w Method Accuracy",
                                             title = " "),
                user_accuracy_theme = theme_grey(),
                user_accuracy_diff_theme = theme_bw(),
                user_accuracy_theme_specs = theme(legend.position = "bottom",
                                                  legend.title = element_blank()))
```

Output:


Difference b/w Method Accuracy
1.00
0.75
0.50
0.25
0.00
3.60E-09
2.060E-02
8.60E-01


Accuracy
1.00
0.75
0.50
0.25
0.00
Triat 1
Triat 2
Triat 3
Method 1
Method 2

5.6 flex_ld_decay()

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.

5.6.1 Arguments

- ld_data: A dataframe containing LD information. Required columns include:

  'CHR_A' - Chromosome identifier for SNP_A.

  'BP_A' - Base pair position of SNP_A.

  'BP_B' - Base pair position of SNP_B.

  'R2' - Linkage disequilibrium (R-squared) value.

  'SNP_A' - Identifier for SNP_A.

  'SNP_B' - Identifier for SNP_B.

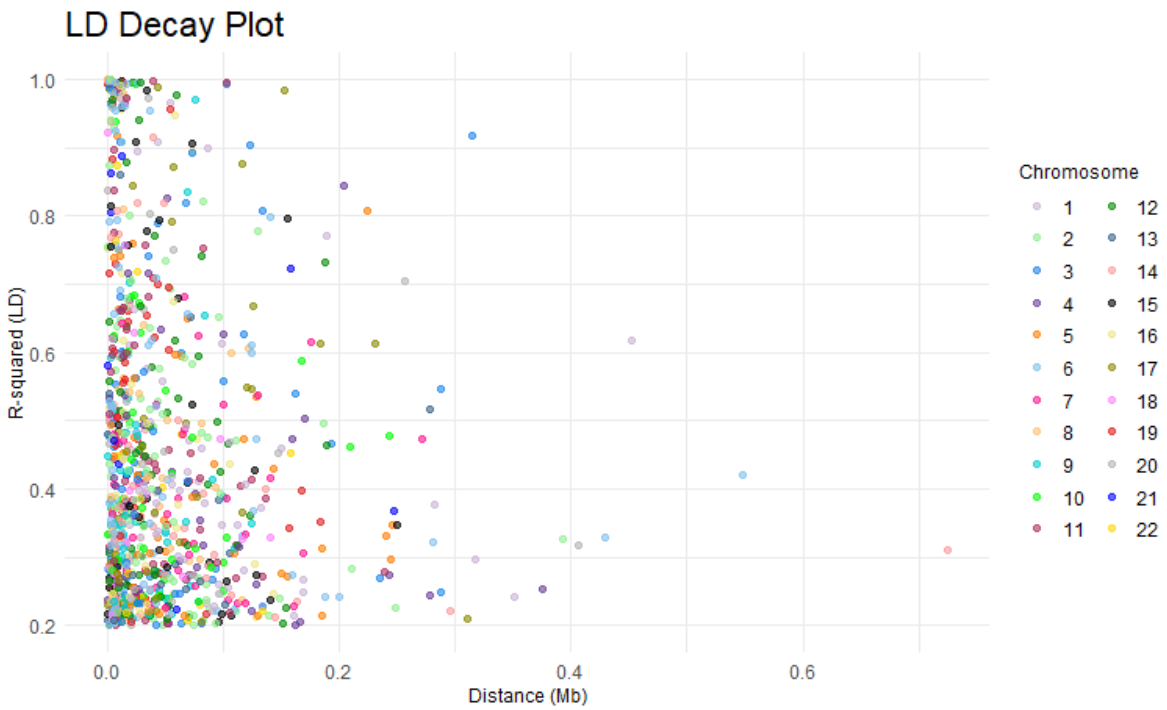
- interactive: Logical. If 'TRUE', returns an interactive plot using 'plotly'. Default is 'TRUE'.
- user_colors: Character vector of colors for chromosomes. If 'NULL', default colors are used. Default is 'NULL'.
- user_title: Character. Title of the plot. Default is '"LD Decay Plot".
- user_x_title: 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". Users can specify one of "lm", "glm", "gam", "loess" or NULL.
- 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.).

5.6.2 Default LD decay plot

Command:

```
flex_LD_decay(example_data7)
```

Output:

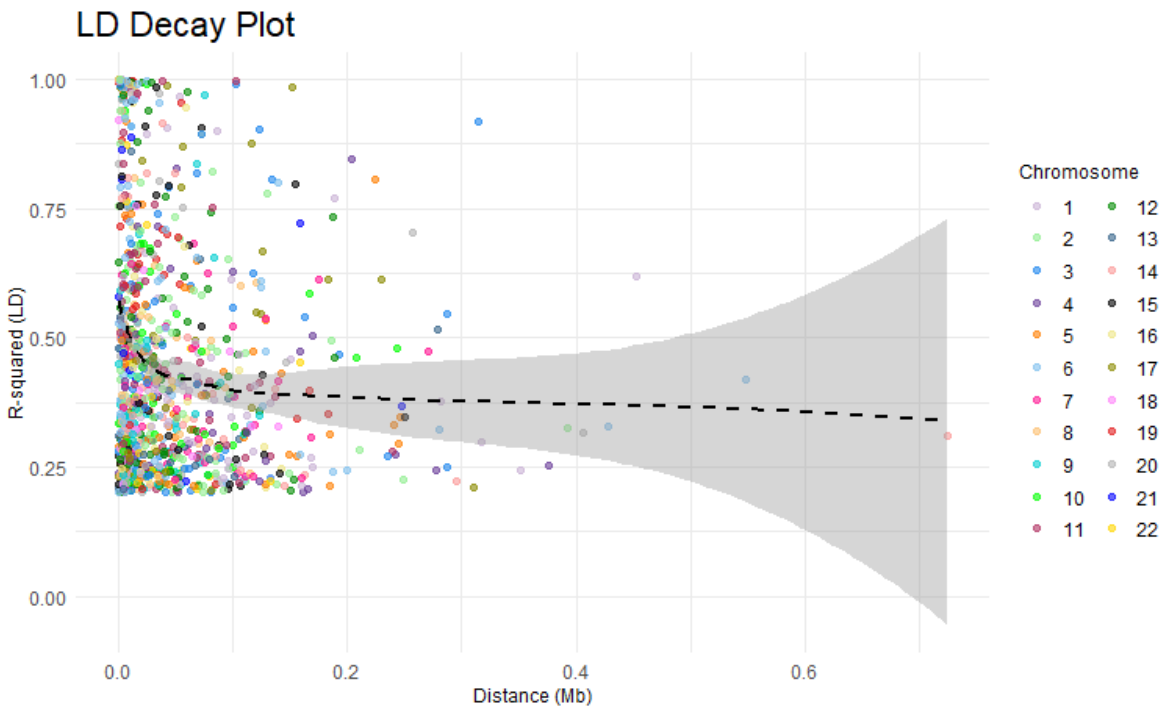


5.6.3 LD decay plot with loess curve

Command:

```
flex_LD_decay(example_data7, add_smoothing = TRUE)
```

Output:

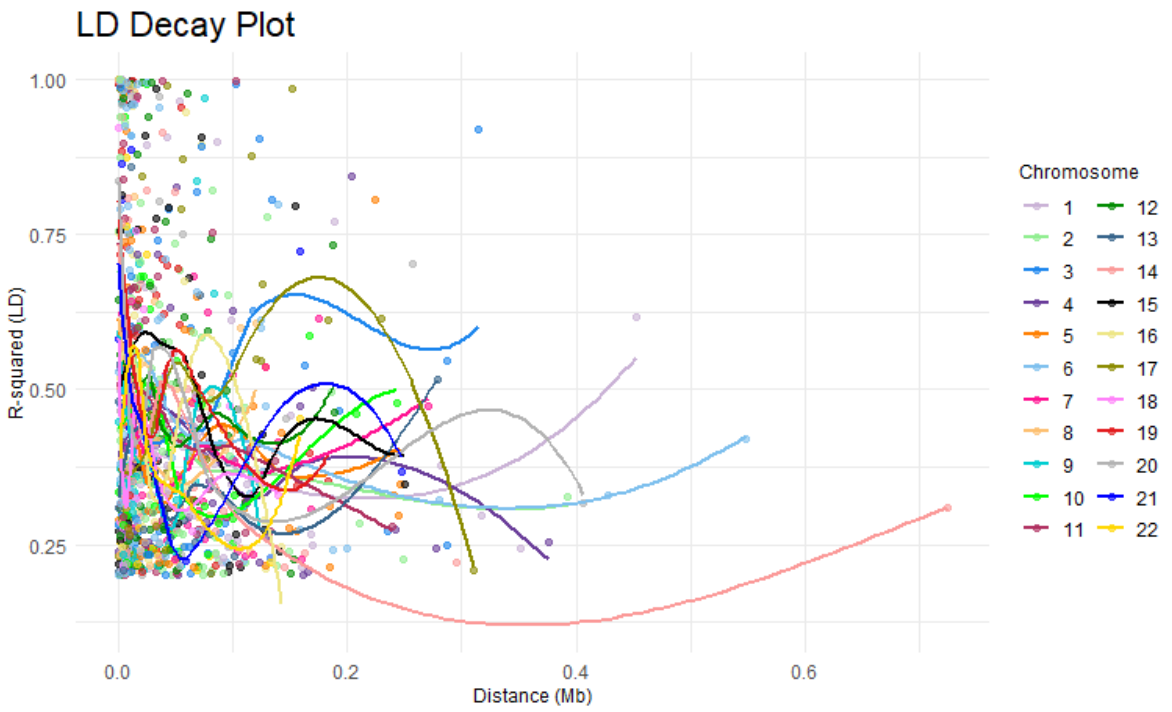


5.6.4 LD decay plot with chromosome specific loess curves

Command:

```
flex_LD_decay(example_data7, add_smoothing = TRUE, add_smoothing_per_chromosome = TRUE)
```

Output:

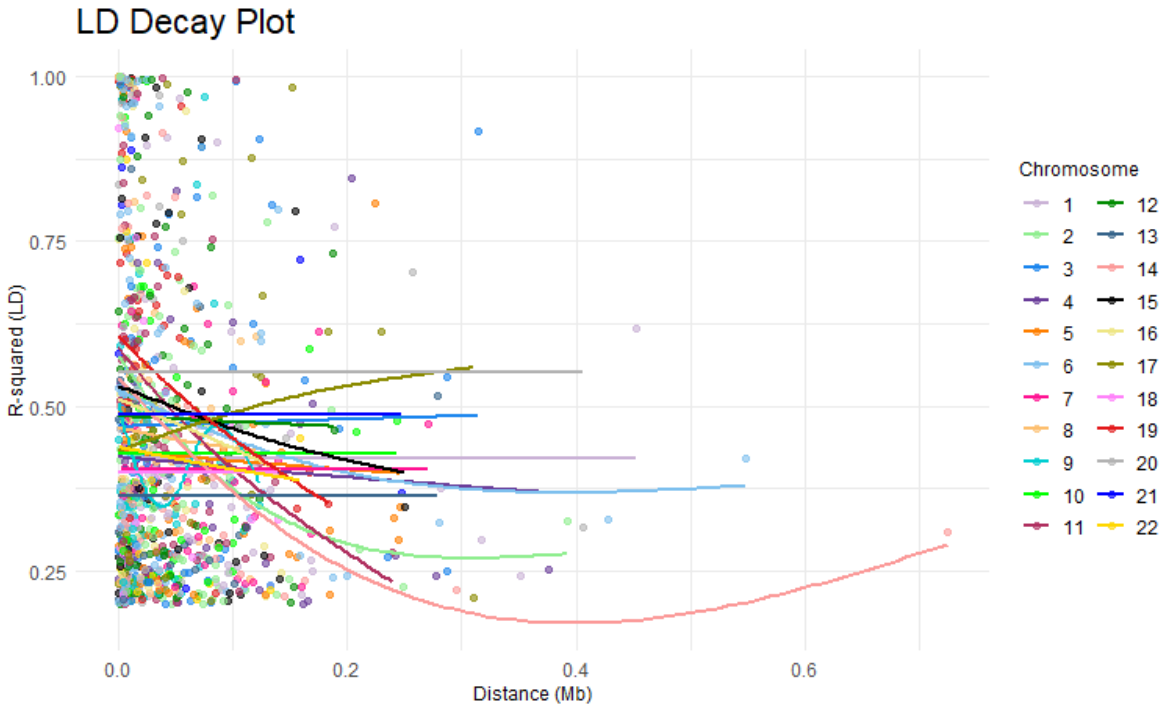


5.6.5 LD decay plot with chromosome specific user-specified (e.g. genaralized additive model) smoothing curves

Command:

```
flex_LD_decay(example_data7, user_smoothing = "gam",
              add_smoothing = TRUE, add_smoothing_per_chromosome = TRUE)
```

Output:

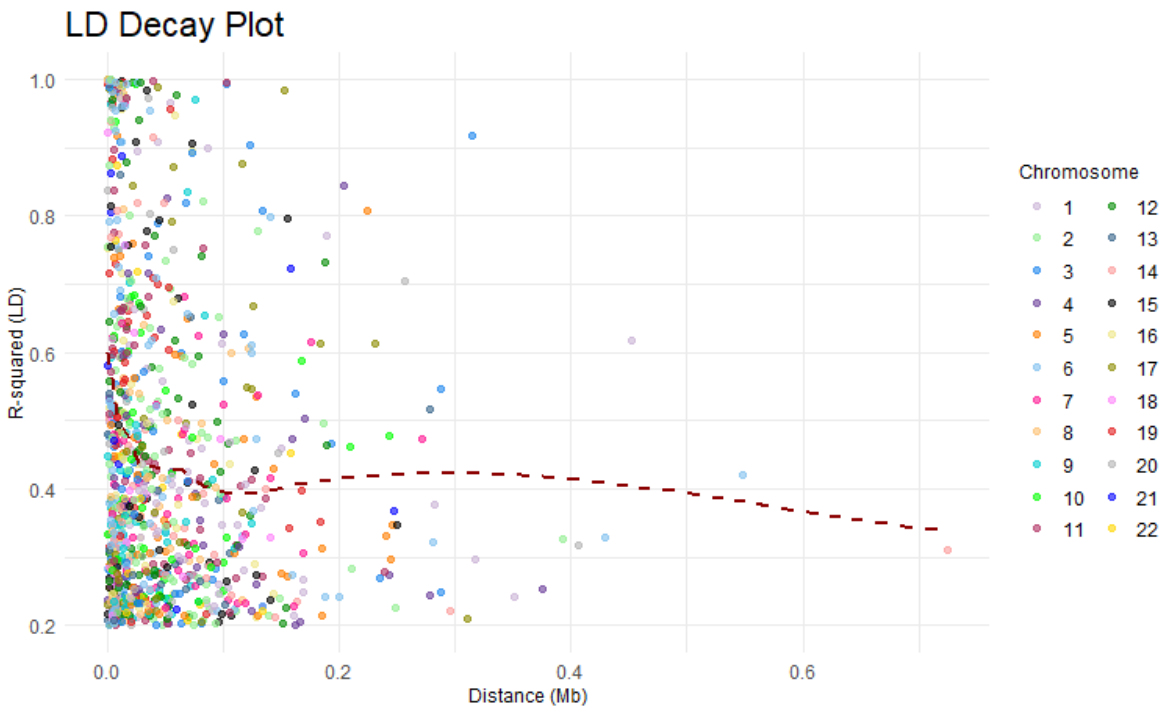


5.6.6 LD decay plot with additional smoothing parameters

Command:

```
flex_LD_decay(example_data7, add_smoothing = TRUE, span = 0.5, se = FALSE, col = "darkred")
```

Output:

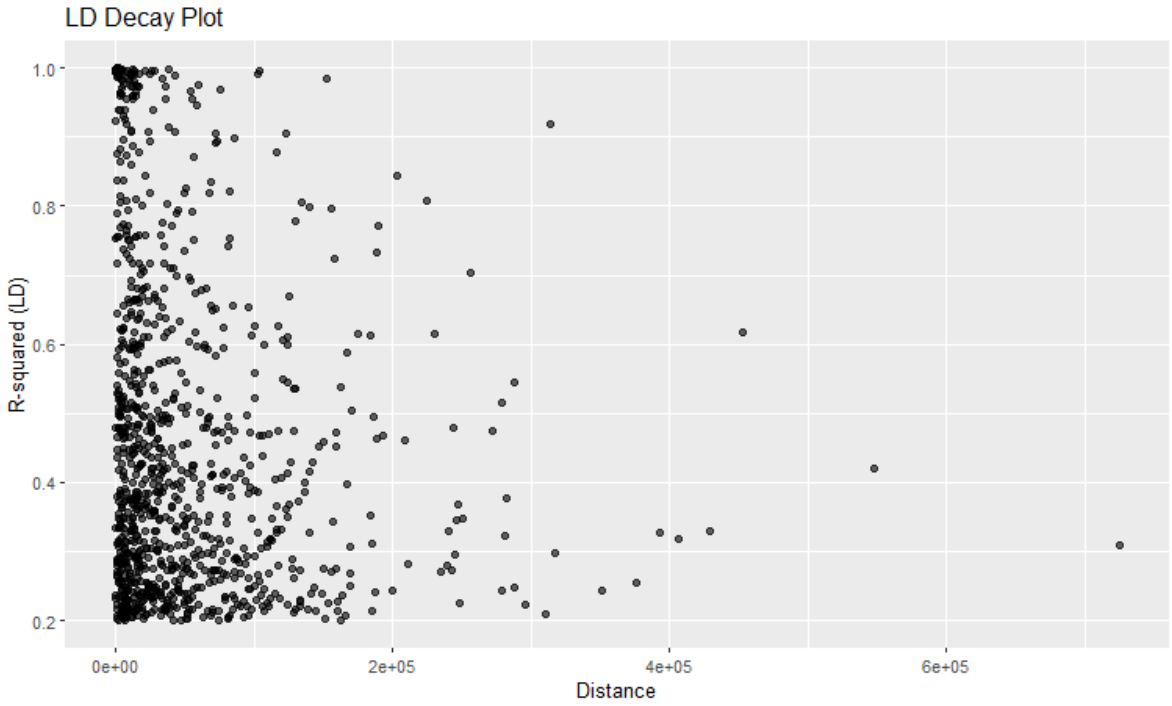


5.6.7 LD decay plot with other user-specified formatting

Command:

```
flex_LD_decay(example_data7, user_colors = rep(1, 22),
              user_plot_theme_specs = theme(legend.position = "none"),
              user_base = 1, user_x_title = "Distance",
              interactive = FALSE, user_plot_theme = theme_grey())
```

Output:



5.7 flex_distribution()

This function generates a customizable 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.

5.7.1 Arguments

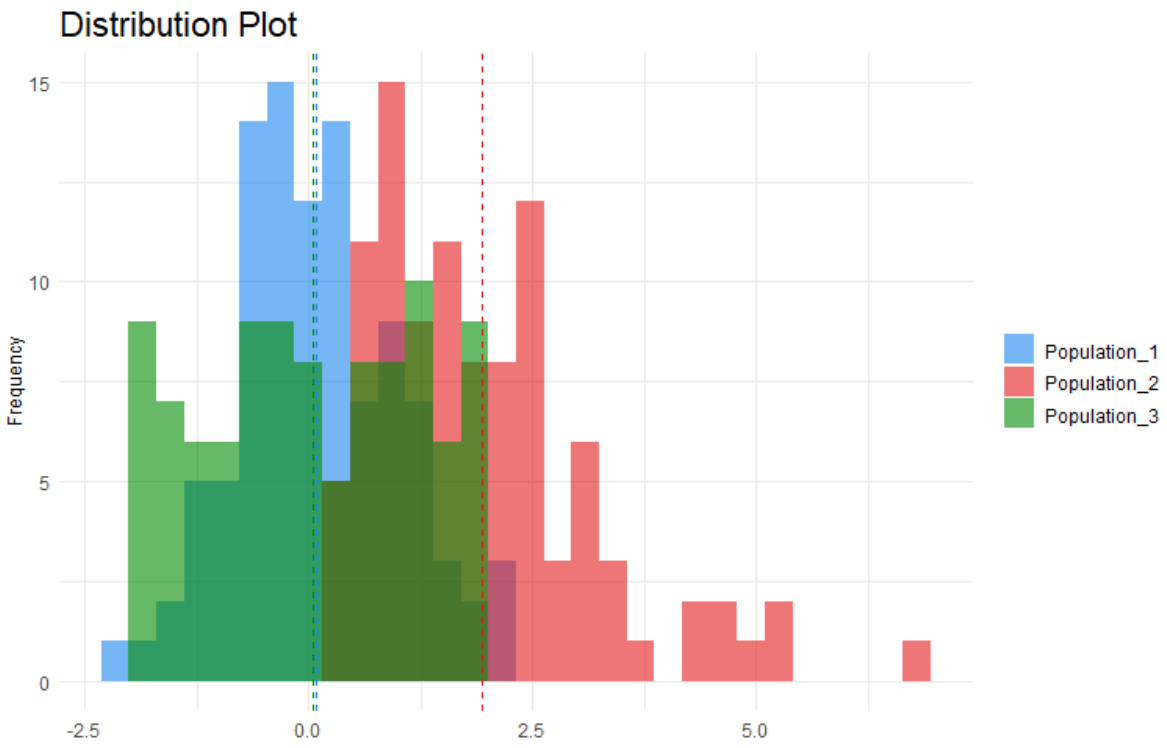
- distribution_data: A dataframe where each column represents a data distribution (e.g. population) to be plotted with minimum of a single column.
- interactive: Logical, if TRUE (default), the plot will be interactive using Plotly, otherwise a ggplot object is returned.
- user_colors: A vector of colors 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).

5.7.2 Default distribution plot

Command:

```
flex_distribution(example_data8)
```

Output:

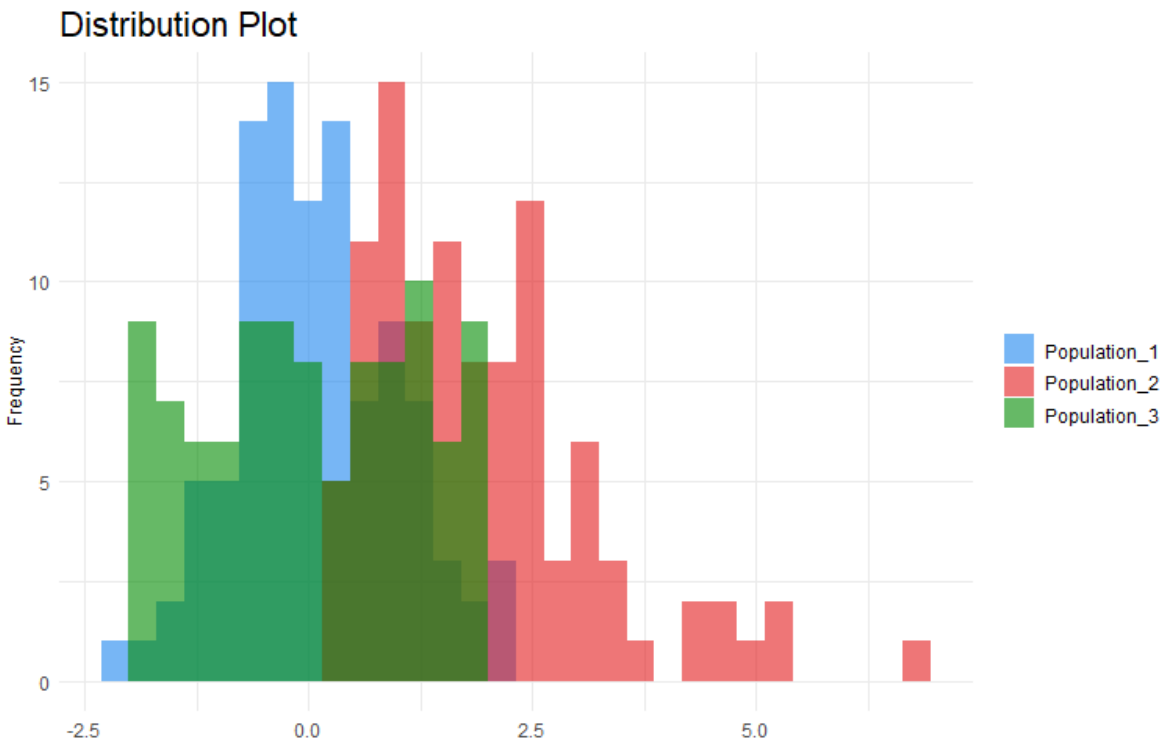


5.7.3 Distribution plot without the default mean reference line(s)

Command:

```
flex_distribution(example_data8, reference_line = FALSE)
```

Output:

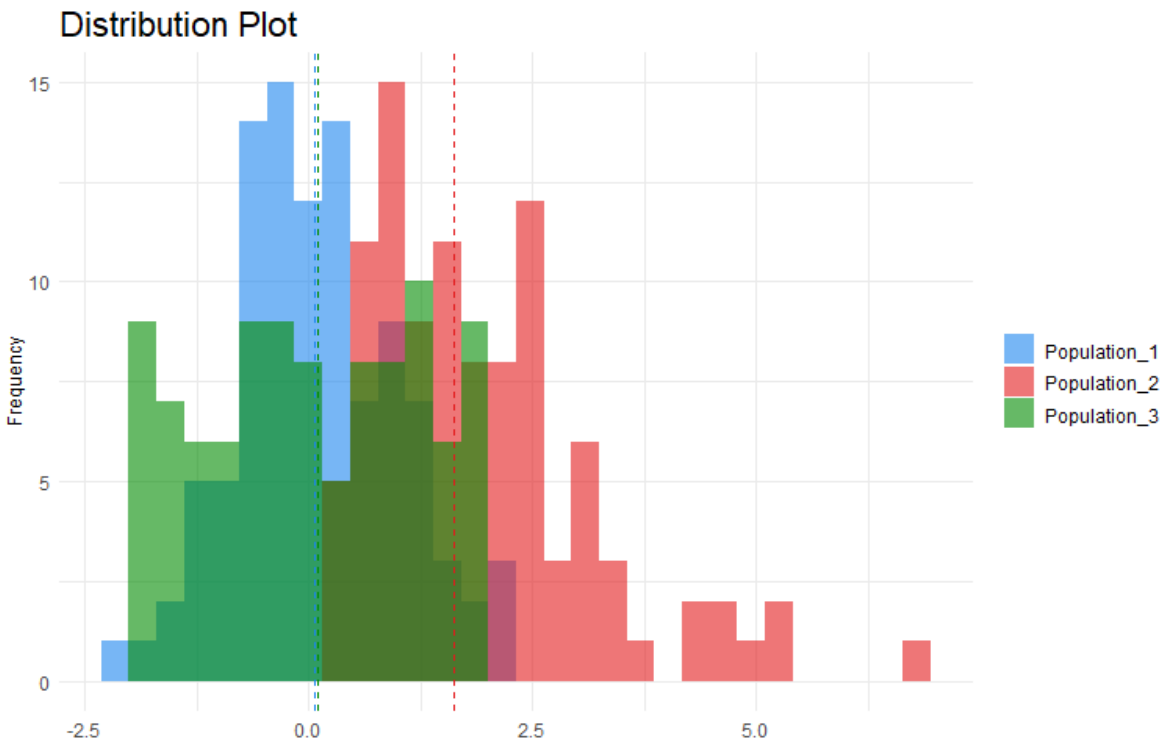


5.7.4 Distribution plot with median as reference line(s)

Command:

```
flex_distribution(example_data8, reference_line = "median")
```

Output:

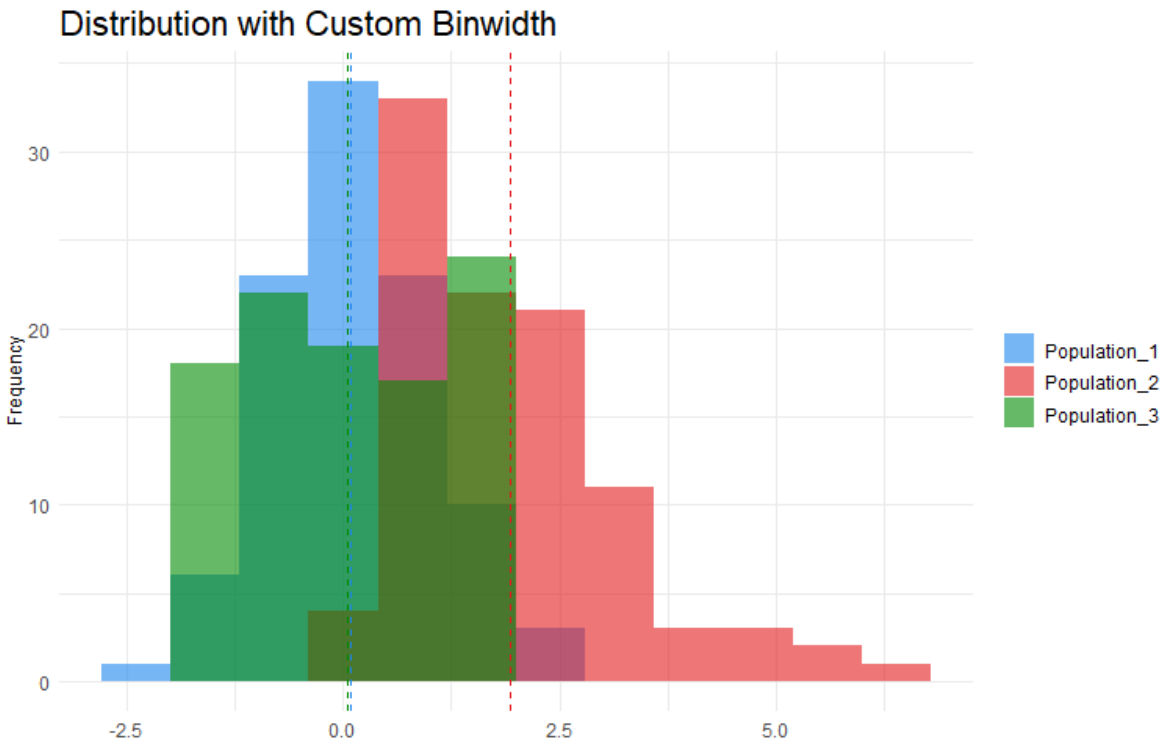


5.7.5 Distribution plot with user-specified binwidth

Command:

```
flex_distribution(example_data8, binwidth = 0.8,
user_title = "Distribution with Custom Binwidth")
```

Output:

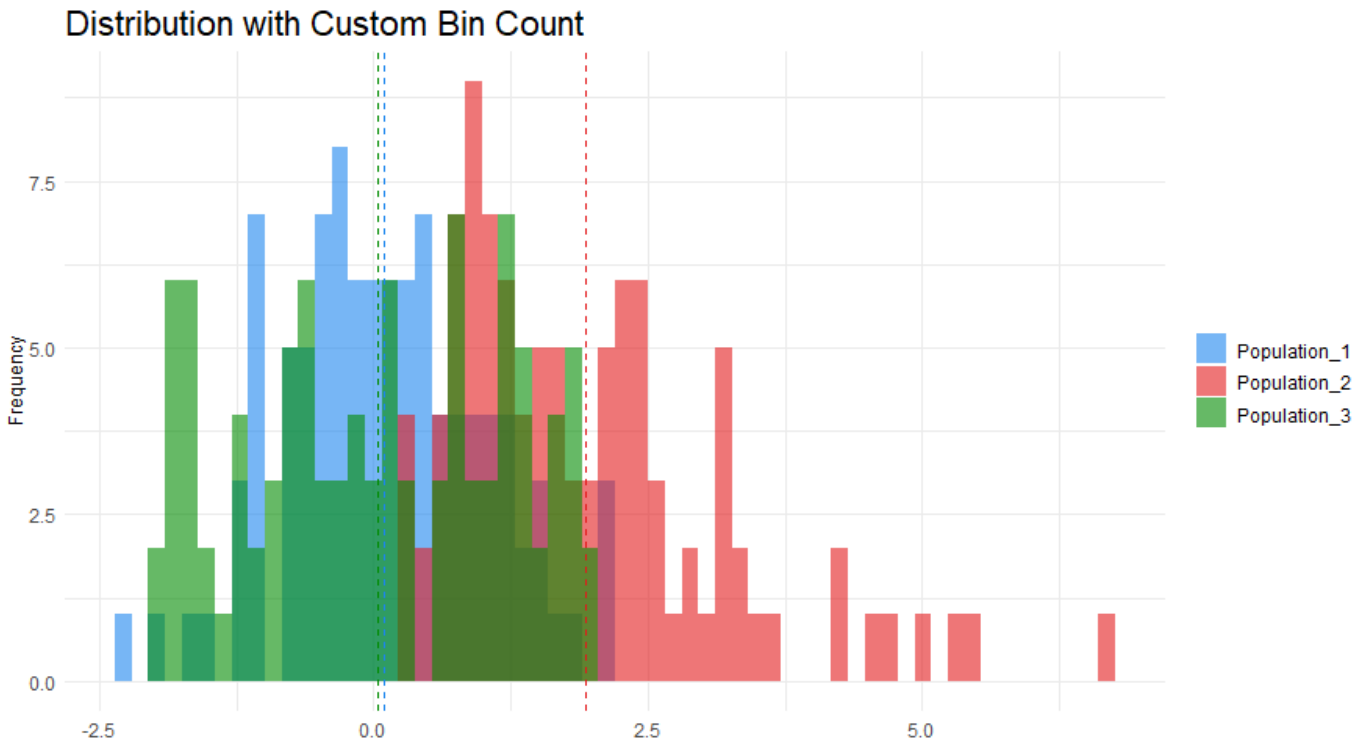


5.7.6 Distribution plot with user-specified number of bins

Command:

```
flex_distribution(example_data8, bins = 60,
user_title = "Distribution with Custom Bin Count")
```

Output:

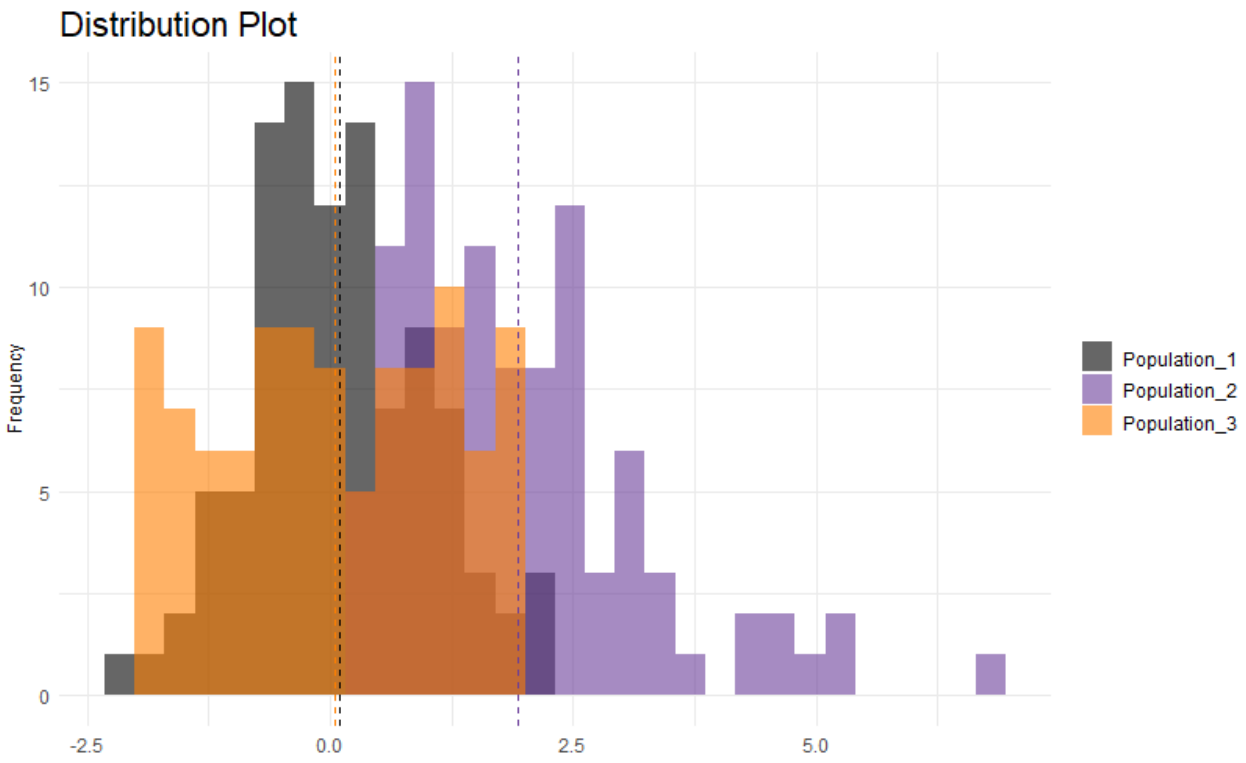


5.7.7 Distribution plot with user-specified colors

Command:

```
flex_distribution(example_data8, user_colors = c("black", "#6A3D9A", "#FF7F00"))
```

Output:

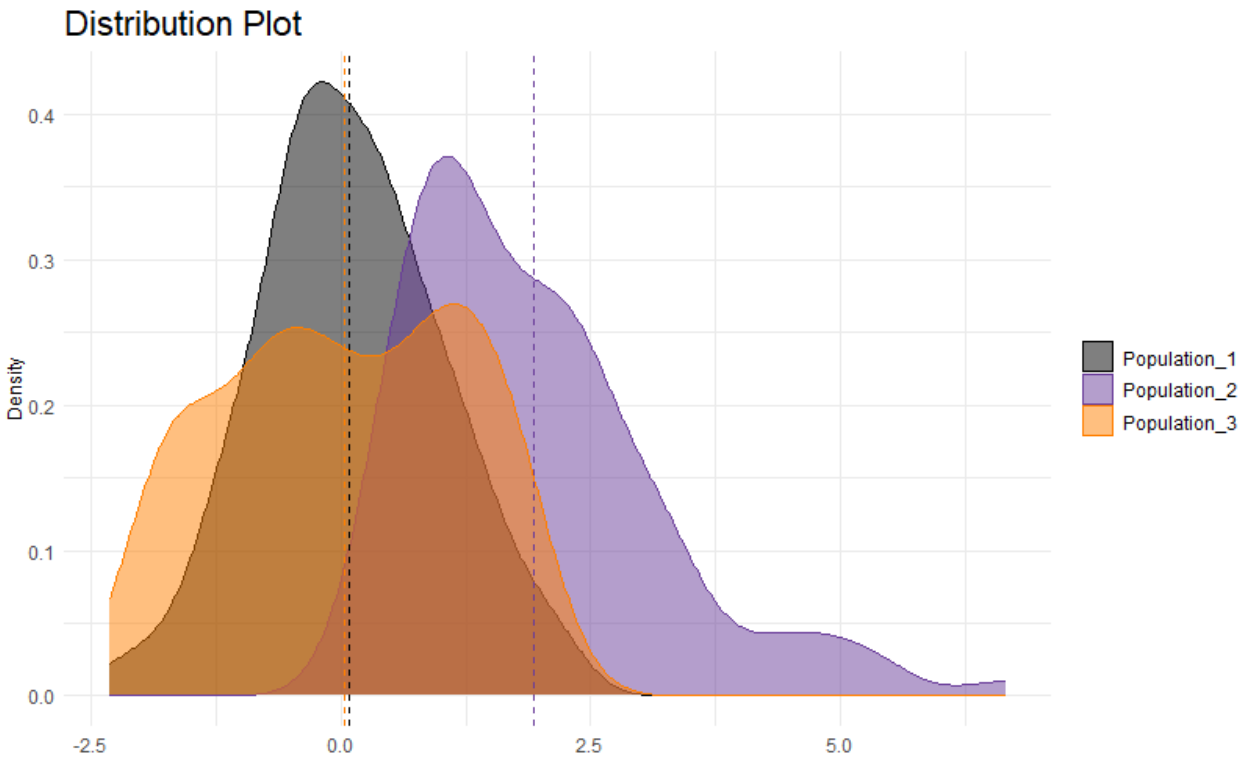


5.7.8 Distribution as (a) density plot/s

Command:

```
flex_distribution(example_data8, plot_type = "density",
user_colors = c("black", "#6A3D9A", "#FF7F00"), user_y_title = "Density")
```

Output:

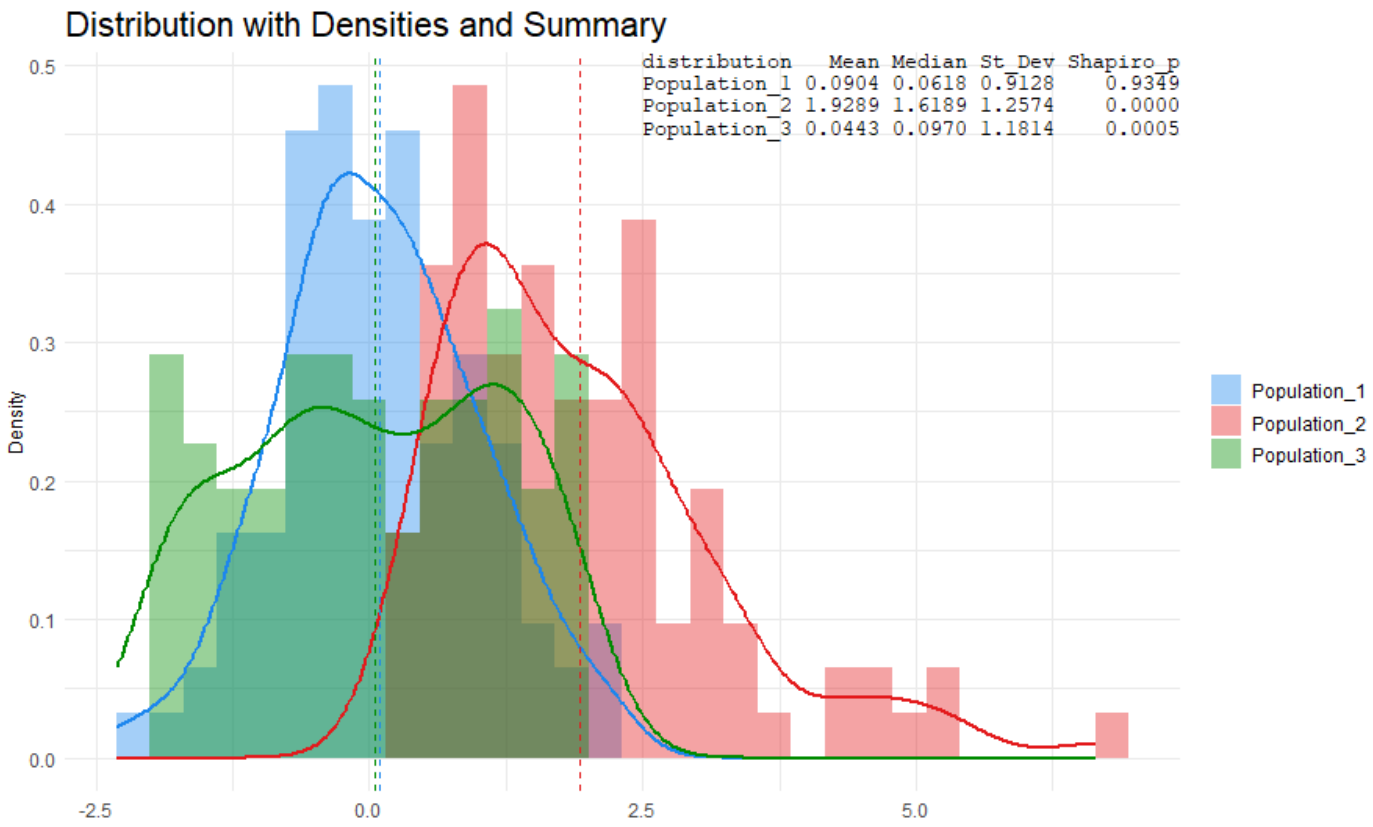


5.7.9 Distribution plot(s) with density curve(s) and summary

Command:

```
    flex_distribution(example_data8, add_density = TRUE,
    show_summary = TRUE, user_title = "Distribution with Densities and Summary",
    user_y_title = "Density"
)
```

Output:



5.8 flex_boxplot()

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.

5.8.1 Arguments

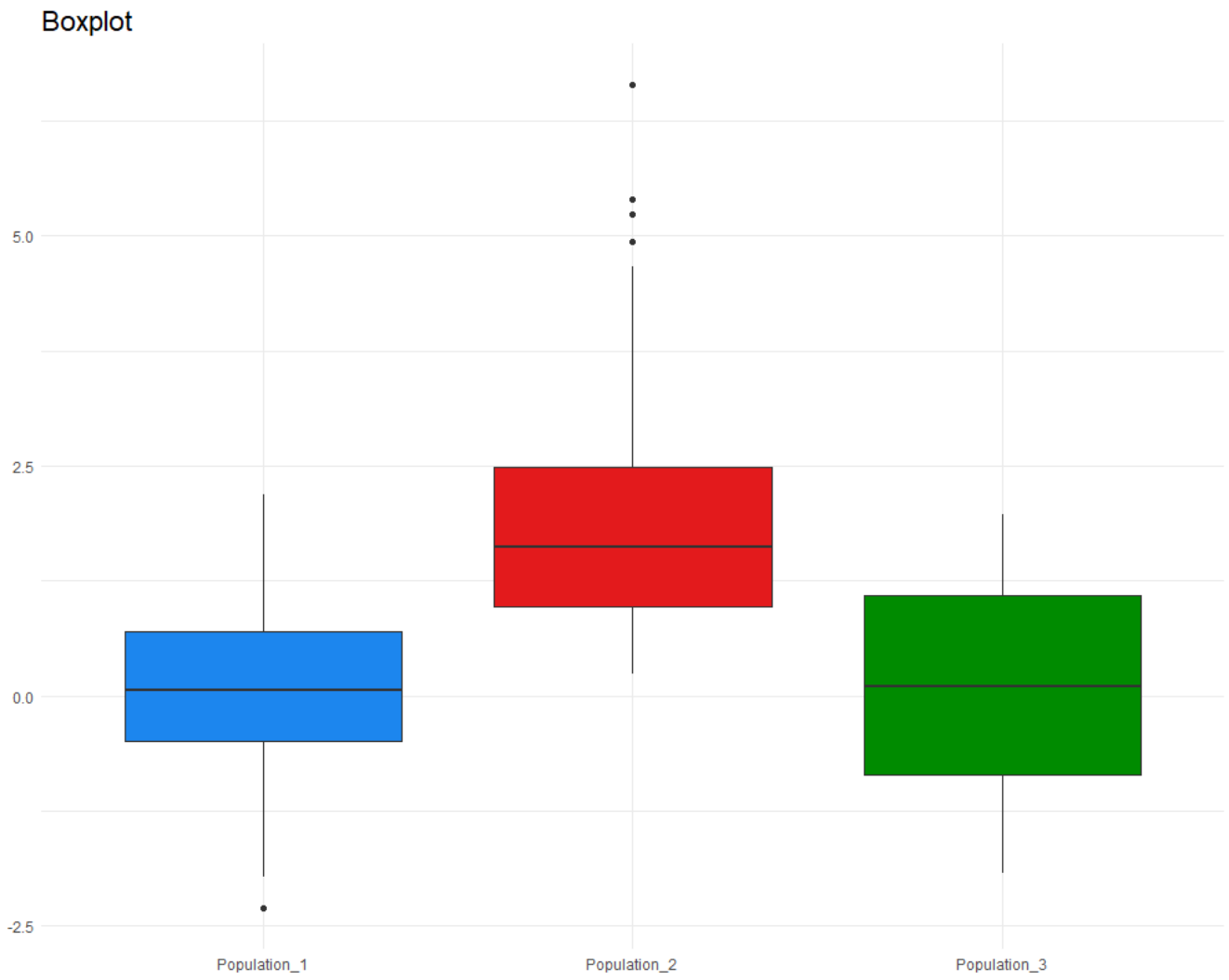
- distribution_data: A dataframe where each column represents a data distribution (e.g. population) to be plotted with minimum of a single column.
- interactive: Logical; if TRUE, the output is an interactive Plotly graph. Defaults to TRUE.
- user_colors: A vector of colors to be used for the boxplots. 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.

5.8.2 Default boxplot

Command:

```
flex_boxplot(example_data8)
```

Output:

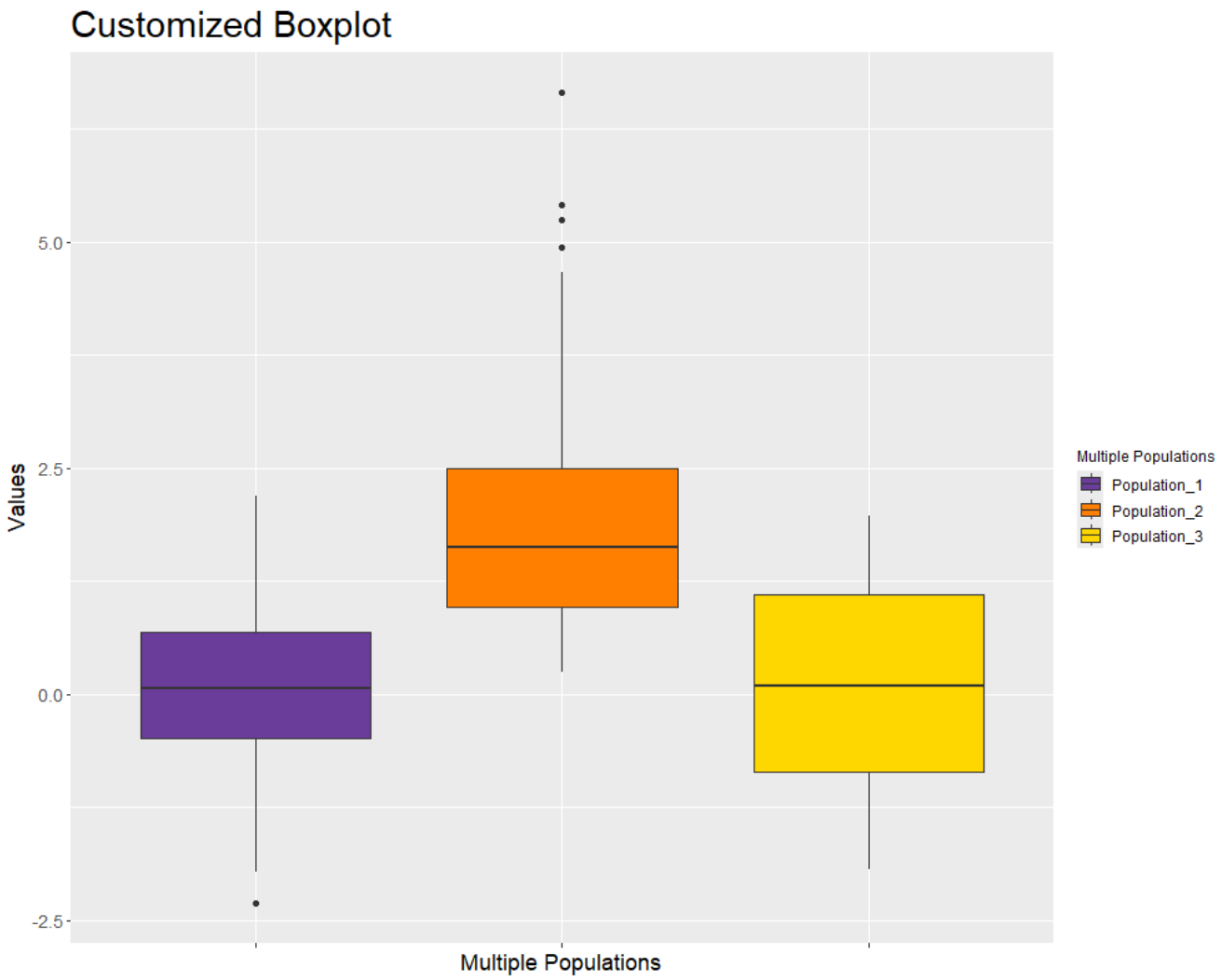


5.8.3 Boxplot with custom colors, themes and titles

Command:

```
flex_boxplot(example_data8, user_plot_theme = 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 = theme(
           legend.title = element_text(size = 10),
           legend.text = element_text(size = 10),
           title = element_text(size = 20),
           axis.text.x = element_blank(),
           axis.title.x = element_text(size = 15),
           axis.text.y = element_text(size = 12),
           axis.title.y = element_text(size = 15) ))
```

Output:

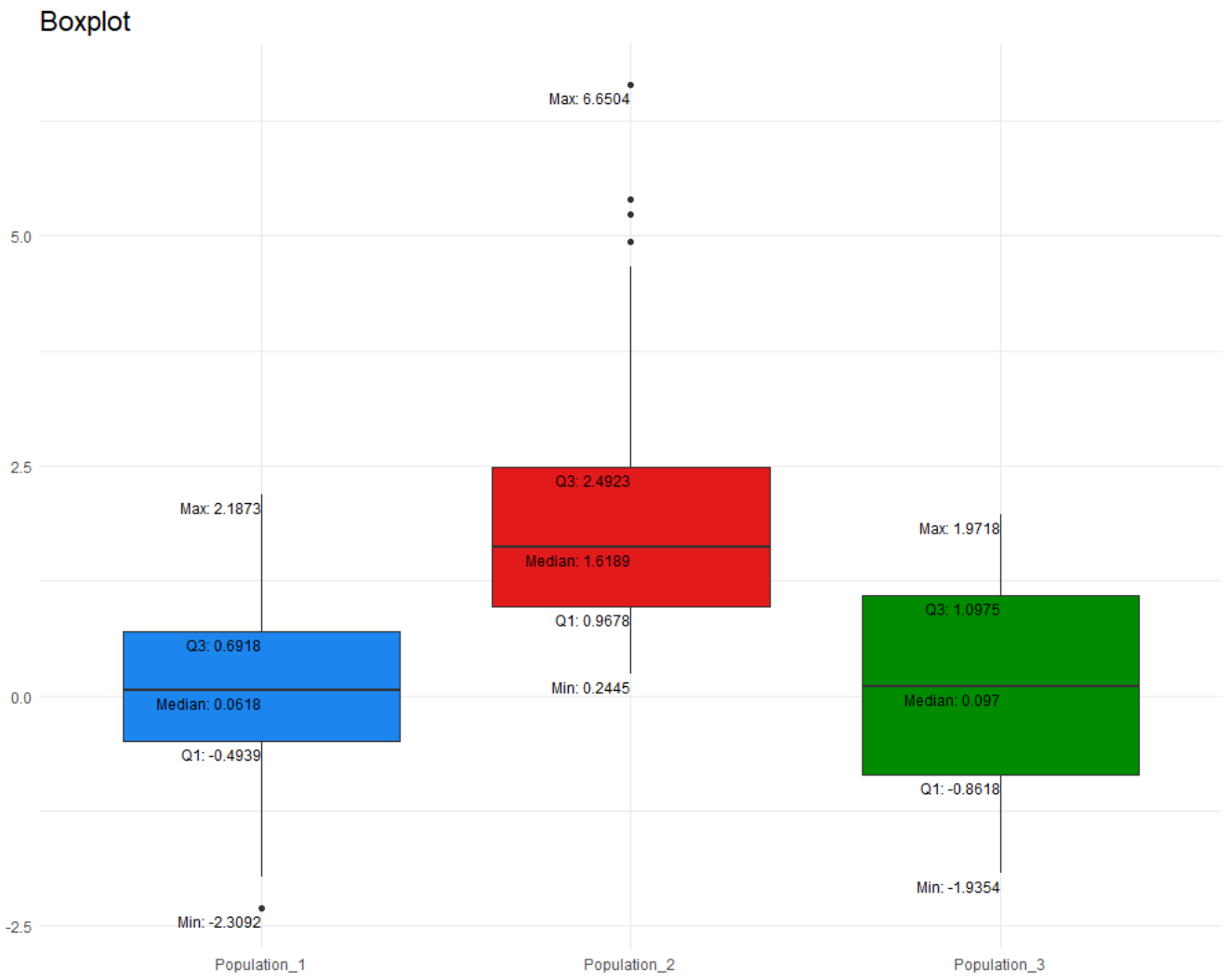


5.8.4 Boxplot with statistics annotations

Command:

```
flex_boxplot(example_data8, annotate_stats = TRUE)
```

Output:

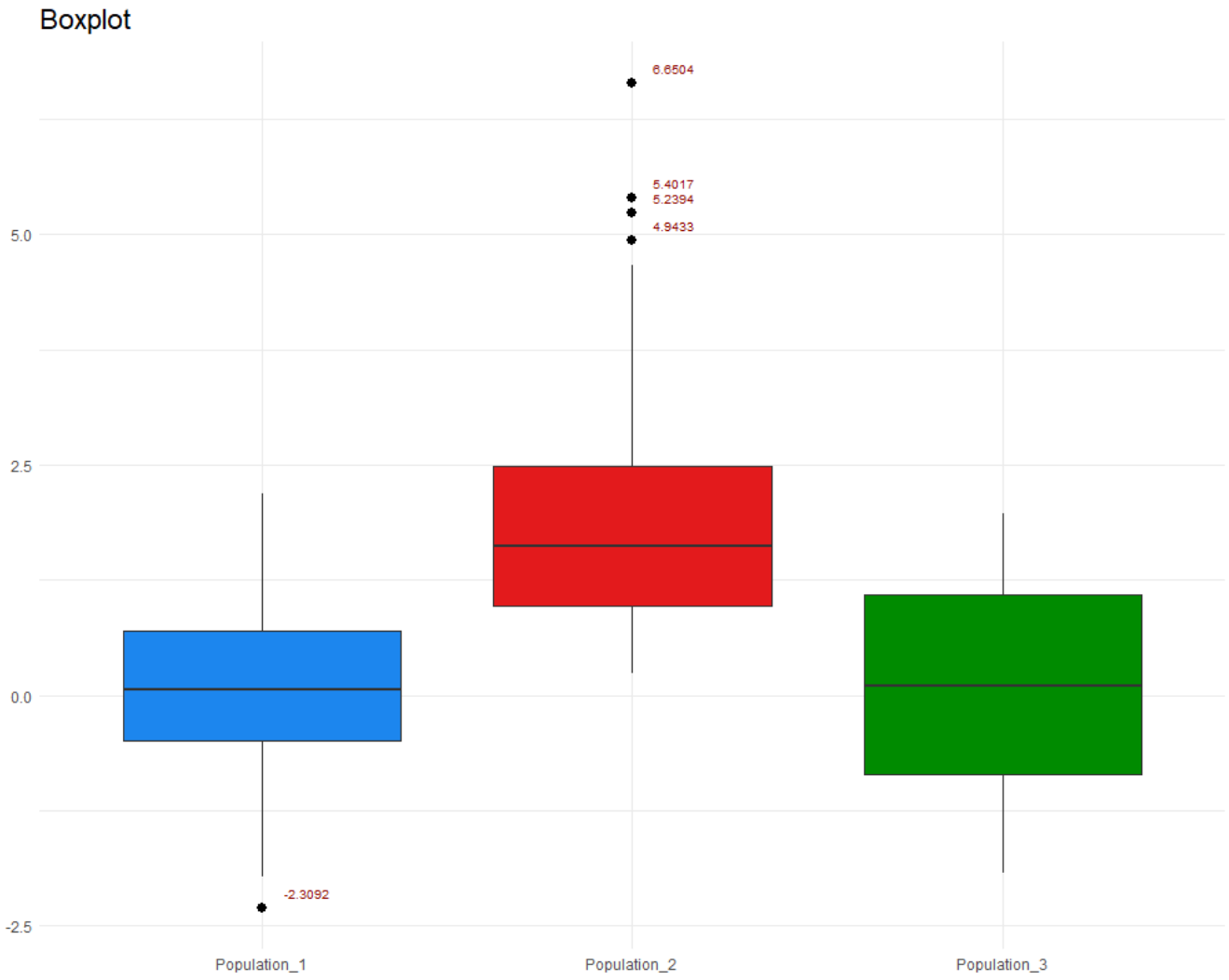


5.8.5 Boxplot with outlier annotations

Command:

```
flex_boxplot(example_data8, annotate_outliers = TRUE)
```

Output:

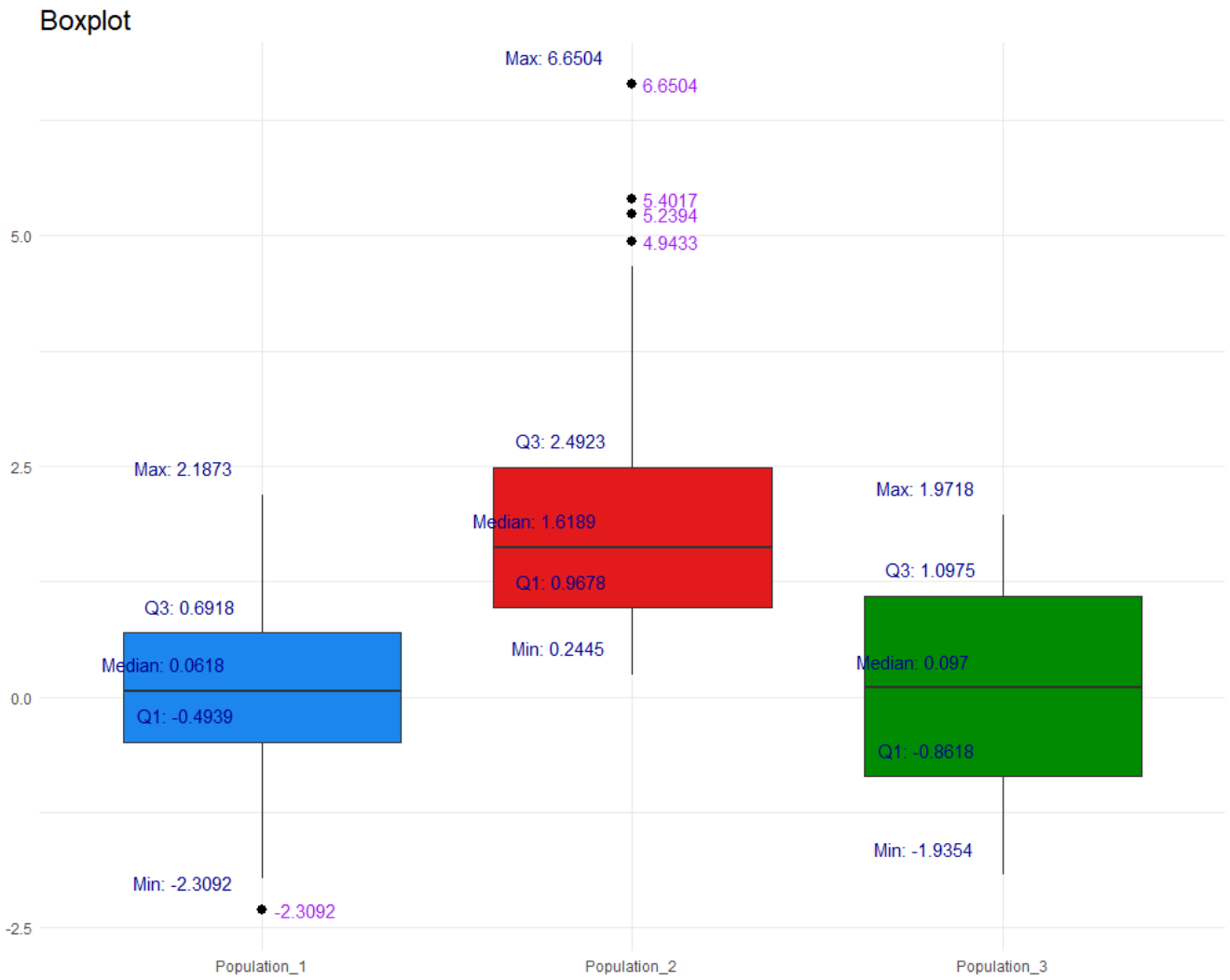


5.8.6 Boxplot with all annotations and customizations

Command:

```
flex_boxplot(example_data8,
         annotate_stats = TRUE,
         annotate_outliers =  TRUE,
         annotate_stats_text_size = 4.5,
         annotate_outliers_text_size = 4.5,
         annotate_stats_text_color = "darkblue",
         annotate_outliers_text_color = "purple",
         annotate_outliers_text_vjust = 0.5,
         annotate_stats_text_vjust = -1.3,
         annotate_outliers_text_hjust = -0.2,
         annotate_stats_text_hjust = 1.3)
```

Output:



5.9 flex_correlation_plot()

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.

5.9.1 Arguments

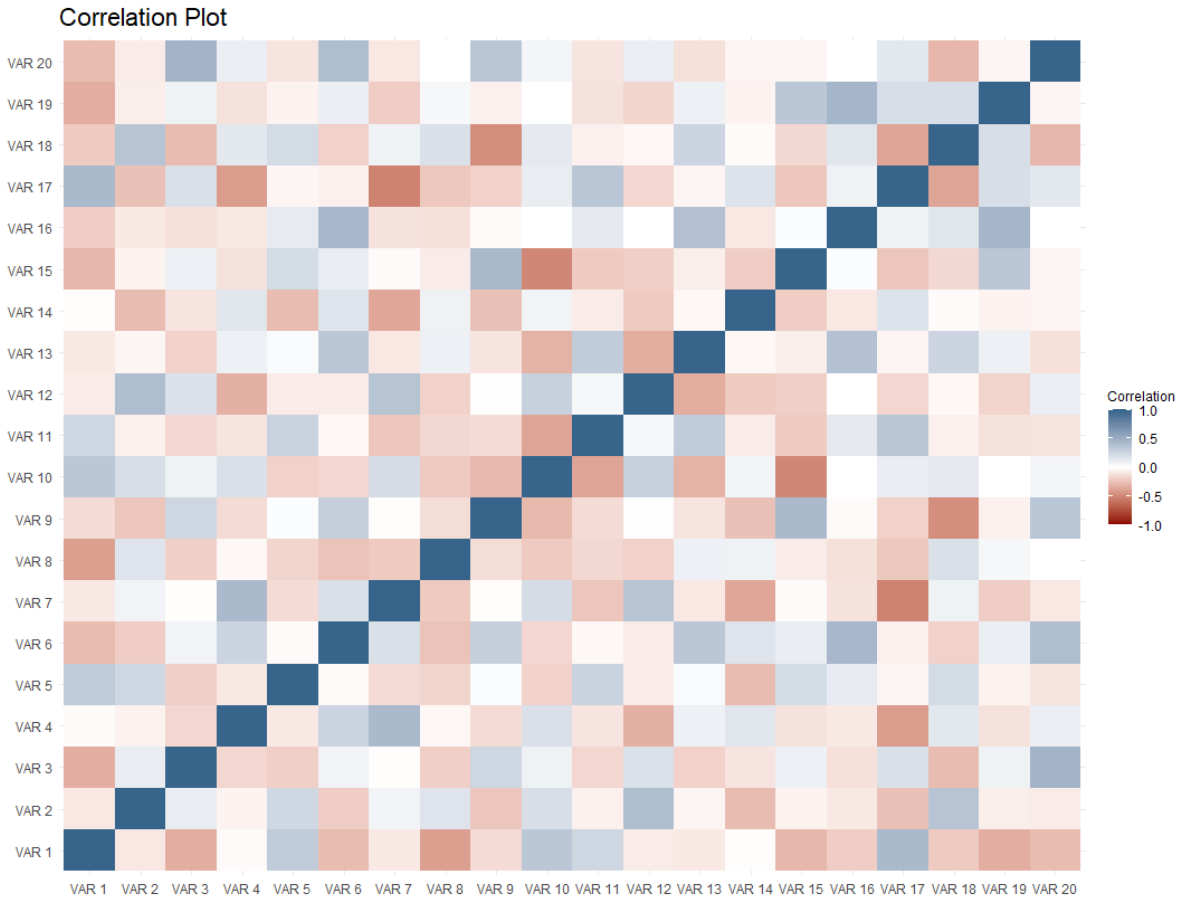
- correlation_matrix: A square matrix representing correlation coefficients with column names identical to row names.
- user_colors: A vector of three colors representing the gradient of the heatmap for negative, neutral, and positive correlations respectively. Default is c("darkred", "white", "steelblue4").
- 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.

5.9.2 Default correlation plot

Command:

```
flex_correlation_plot(example_data9)
```

Output:

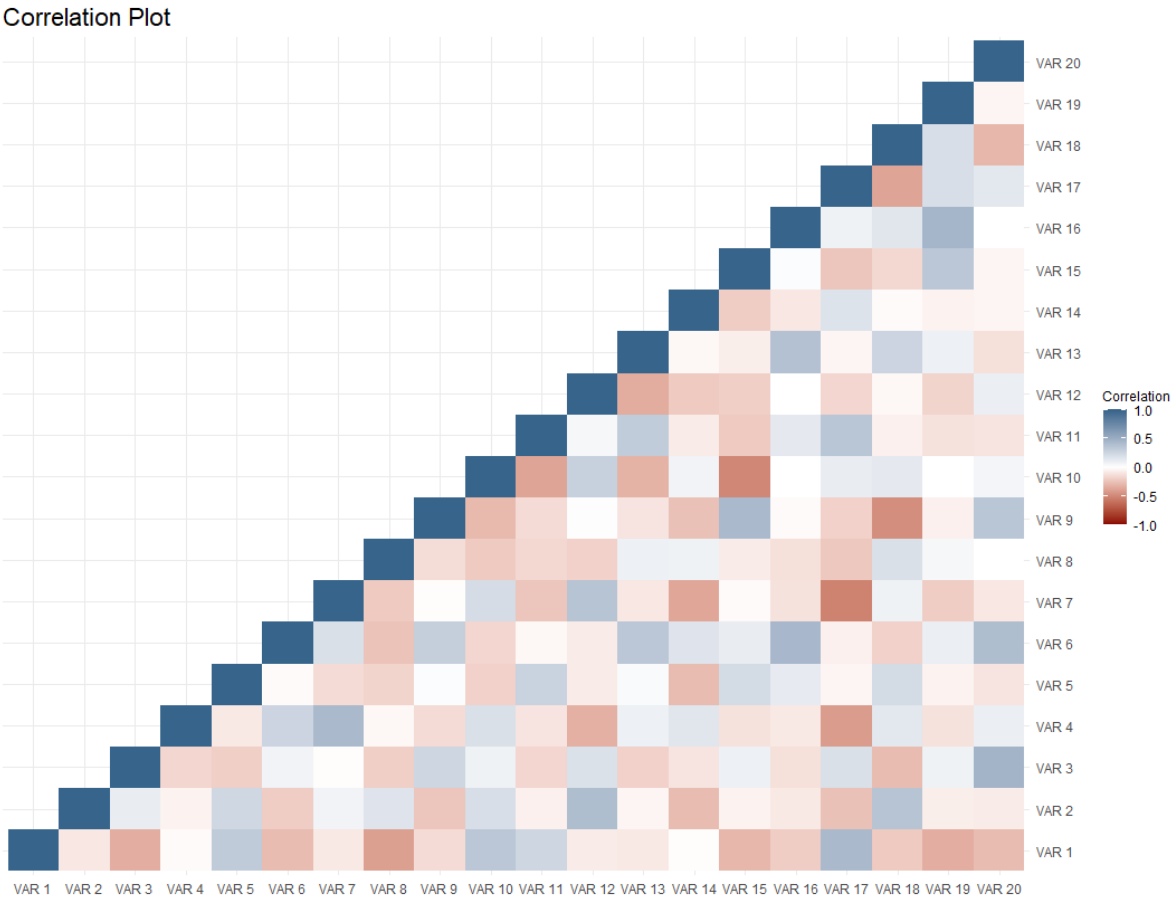


5.9.3 Correlation plot displaying lower triangle only

Command:

```
flex_correlation_plot(example_data9, matrix_type = "lower")
```

Output:

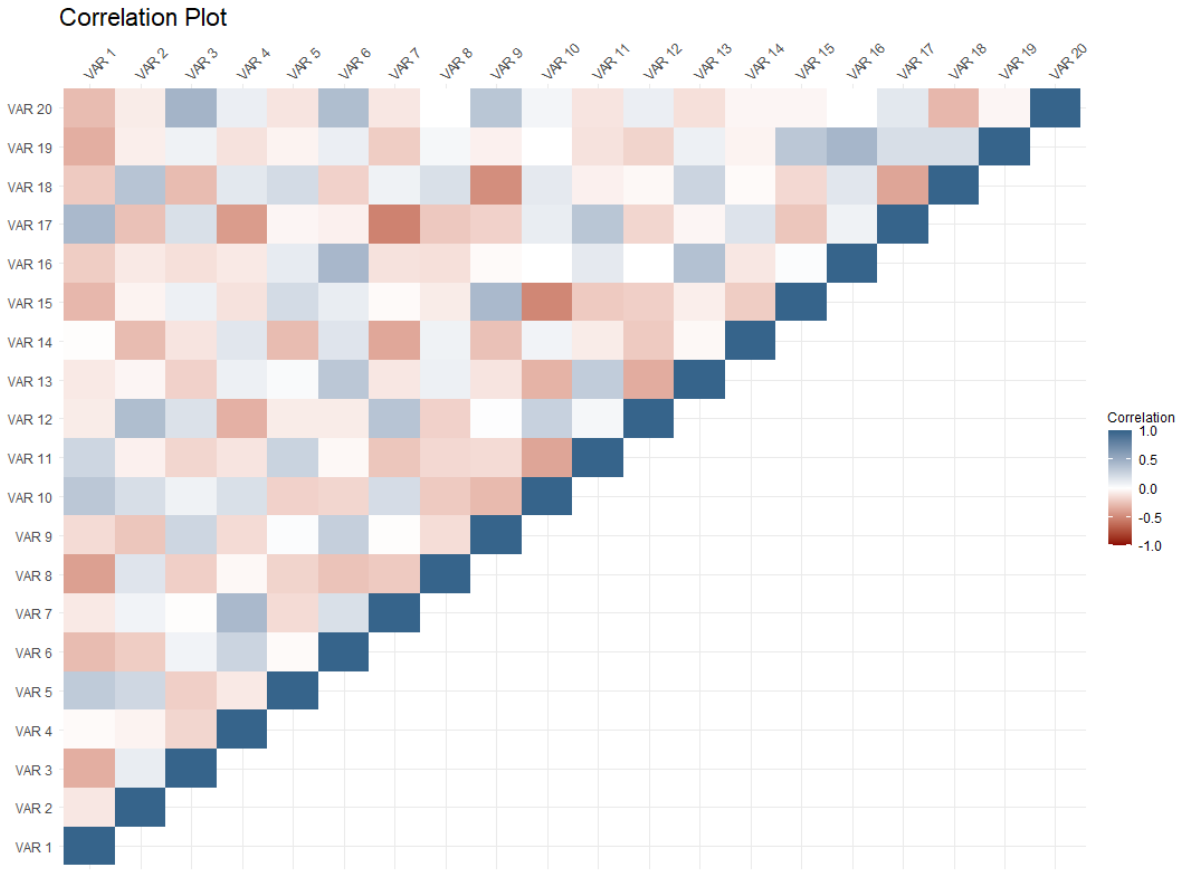


5.9.4 Correlation plot displaying upper triangle only

Command:

```
flex_correlation_plot(example_data9, matrix_type = "upper")
```

Output:

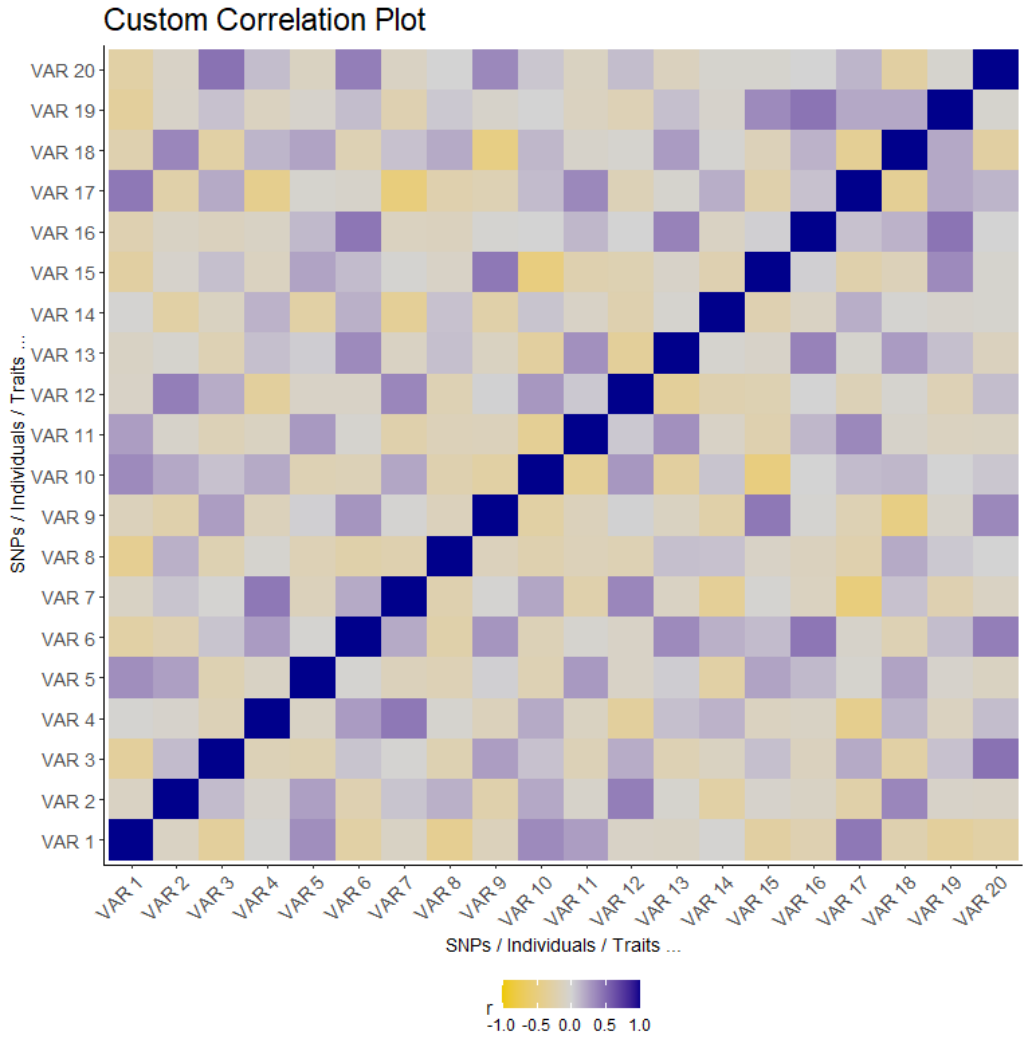


5.9.5 Correlation plot with user-specified formatting

Command:

```
  flex_correlation_plot(
correlation_matrix = example_data9,
user_colors = c("gold2", "lightgrey", "darkblue"),
display_names = TRUE, user_title = "Custom Correlation Plot",
user_x_title = "SNPs / Individuals / Traits ...",
user_y_title = "SNPs / Individuals / Traits ...",
user_legend_title = "r", 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, vjust = 1),
  axis.text.y = element_text(size = 12),
  axis.title.y = element_text(size = 12),
  legend.position = "bottom"))
```

Output:

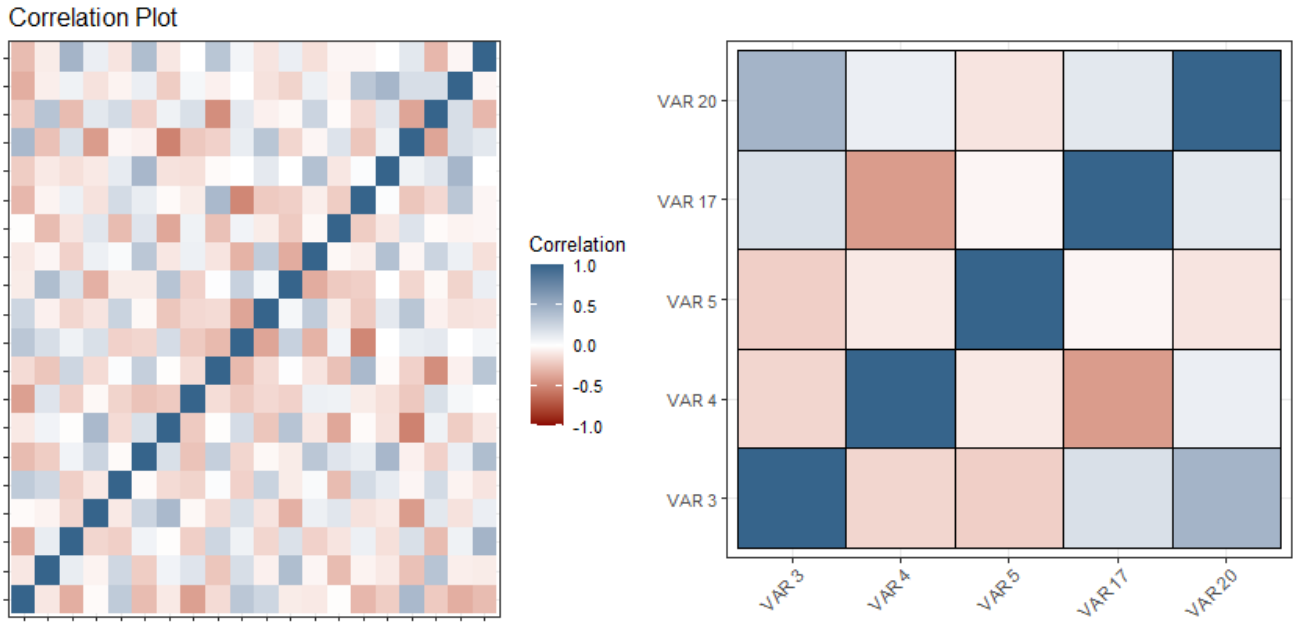


5.9.6 Correlation plot with user-specified zoomed-in range

Command:

```
flex_correlation_plot(example_data9,
                      user_zoom_range = c(3:5, 17, 20),
                      user_plot_theme = theme_bw(),
                      user_plot_theme_specs = theme(axis.text = element_blank()))
```

Output:



5.10 flex_regression_summary()

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.

5.10.1 Arguments

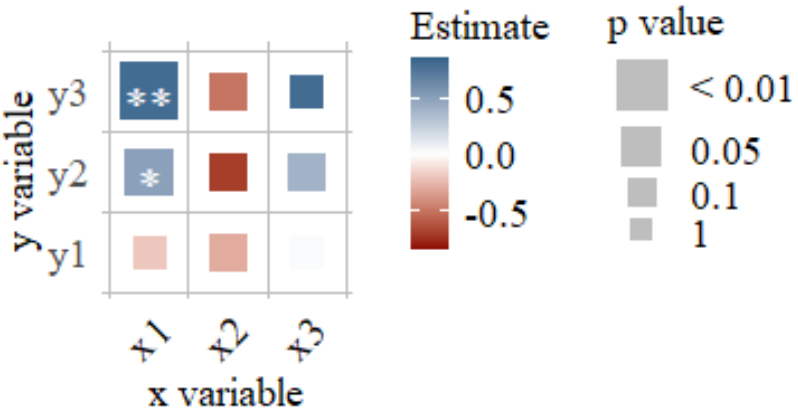
- regression_data: A dataframe containing the variables 'x_var', 'y_var', 'estimate', and 'p_value' which represent the horizontal axis component, vertical axis component, regression coefficient estimates, and their corresponding p-values, respectively.
- interactive: Logical, if TRUE returns an interactive 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.

5.10.2 Default regression summary plot

Command:

```
flex_regression_summary(example_data10)
```

Output:

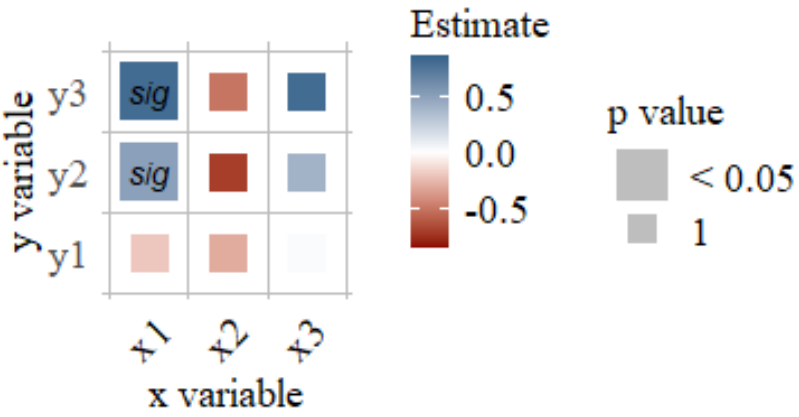


5.10.3 Regression summary plot with user-specific p value break points and related text annotations

Command:

```
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")))
```

Output:



5.10.4 Regression summary plot with user-specific formatting

Command:

```
flex_regression_summary(example_data10,
                        user_geom_point = geom_point(shape = 25, color = "white", stroke = 0),
                        user_gradient_bar = scale_fill_gradientn(colours = c("purple4", "white",
                        "green4"), limit = c(-max(abs(example_data10$estimate))-0.05,
                        max(abs(example_data10$estimate))+0.05),
                        guide = guide_colorbar(title = "user estimate lab",
                        label.position = "right", barwidth = 1, barheight = 5,
                        title.position = "top", order = 1)),
                        user_theme_specs = theme(axis.text.x = element_text(angle = 0,
                                                                            hjust = 1,
                                                                            vjust = 1,
                                                                            size = 15,
                                                                            colour = "darkred")),
                        user_labs = labs(x = "user x lab", y = "user y lab",
                        fill = "user estimate lab", size = "user p value lab"),
                        user_size_legend_guides = guides(size = guide_legend(reverse = F,
                                                         label.position = "right",
                                                         title.position = "top",
                                                         override.aes = list(colour="darkred",
                                                         shape=25, fill="darkblue"))))
```

Output: