Package 'ggmanh'

November 14, 2025

Title Visualization Tool for GWAS Result

Version 1.15.0

Description Manhattan plot and QQ Plot are commonly used to visualize the end result of Genome Wide Association Study.
The ``ggmanh" package aims to keep the generation of these plots simple while maintaining customizability.
Main functions include manhattan_plot, qqunif, and thinPoints.

biocViews Visualization, GenomeWideAssociation, Genetics

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports gdsfmt, ggrepel, grDevices, paletteer, RColorBrewer, rlang, scales, SeqArray (>= 1.32.0), stats, tidyr, dplyr, pals, magrittr

Depends methods, ggplot2

Suggests BiocStyle, rmarkdown, knitr, testthat (>= 3.0.0), GenomicRanges

Config/testthat/edition 3

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/ggmanh

git_branch devel

git_last_commit 3e4cc5e

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-11-13

Author John Lee [aut, cre], John Lee [aut] (AbbVie), Xiuwen Zheng [ctb, dtc]

Maintainer John Lee <swannyy.stat@gmail.com>

Contents

ggmanh-package 2
binned_manhattan_plot 3
binned_manhattan_preprocess 7
calc_new_pos 10
default_gds_path 11
gds_annotate 11
ggmanh 12
ggmanh_annotation_gds 13
manhattan_data_preprocess 14
manhattan_plot 17
qqunif 21
thinPoints 22

Index **24**

---

ggmanh-package *ggmanh: Visualization Tool for GWAS Result*

Description

Manhattan plot and QQ Plot are commonly used to visualize the end result of Genome Wide Association Study. The "ggmanh" package aims to keep the generation of these plots simple while maintaining customizability. Main functions include manhattan_plot, qqunif, and thinPoints.

Author(s)

Maintainer: John Lee <swannyy.stat@gmail.com>

Authors:

- John Lee <john.lee@abbvie.com> (AbbVie)

Other contributors:

- Xiuwen Zheng <xiuwen.zheng@abbvie.com> [contributor, data contributor]

`binned_manhattan_plot` *Plot Binned Manhattan Plot*

Description

Contrary to the traditional manhattan plot, which plots all the points, the binned manhattan plot vertically and horizontally bins the variants into blocks. This speeds up plotting and reduces clutter in the case of high number of variants. The colors of the blocks can also be used to summarise the variants within each block and highlight certain features.

Usage

```
binned_manhattan_plot(x, ...)

## Default S3 method:
binned_manhattan_plot(x, ...)

## S3 method for class 'MPdataBinned'
binned_manhattan_plot(
  x,
  outfn = NULL,
  signif.lwd = 1,
  bin.outline = FALSE,
  bin.outline.alpha = 0.2,
  highlight.colname = NULL,
  highlight.counts = TRUE,
  bin.palette = "viridis::plasma",
  bin.alpha = 0.9,
  palette.direction = 1,
  nonsignif.default = NULL,
  show.legend = TRUE,
  legend.title = NULL,
  background.col = c("grey90", "white"),
  background.alpha = 0.7,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  ...
)

## S3 method for class 'data.frame'
binned_manhattan_plot(
  x,
  bins.x = 10,
  bins.y = 100,
```

```
  chr.gap.scaling = 0.4,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.colname = "chr",
  pos.colname = "pos",
  chr.order = NULL,
  signif.col = NULL,
  preserve.position = TRUE,
  pval.log.transform = TRUE,
  summarise.expression.list = NULL,
  outfn = NULL,
  signif.lwd = 1,
  bin.outline = FALSE,
  bin.outline.alpha = 0.2,
  highlight.colname = NULL,
  highlight.counts = TRUE,
  bin.palette = "viridis::plasma",
  bin.alpha = 0.9,
  palette.direction = 1,
  nonsignif.default = NULL,
  show.legend = TRUE,
  legend.title = NULL,
  background.col = c("grey90", "white"),
  background.alpha = 0.7,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  ...
)

## S4 method for signature 'GRanges'
binned_manhattan_plot(
  x,
  bins.x = 10,
  bins.y = 100,
  chr.gap.scaling = 0.4,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.order = NULL,
  signif.col = NULL,
  preserve.position = TRUE,
  pval.log.transform = TRUE,
  summarise.expression.list = NULL,
  outfn = NULL,
  signif.lwd = 1,
  bin.outline = FALSE,
```

```
  bin.outline.alpha = 0.2,
  highlight.colname = NULL,
  highlight.counts = TRUE,
  bin.palette = "viridis::plasma",
  bin.alpha = 0.9,
  palette.direction = 1,
  nonsignif.default = NULL,
  show.legend = TRUE,
  legend.title = NULL,
  background.col = c("grey90", "white"),
  background.alpha = 0.7,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  ...
)
```

Arguments

x	a data.frame or any other extension of a data frame. It can also be a MPdata object.
...	Ignored
outfn	a character. File name to save the Manhattan Plot. If outfn is supplied (i.e. !is.null(outfn)), then the plot is not drawn in the graphics window.
signif.lwd	a number. Line width of the significance threshold line.
bin.outline	a logical. Outline each bin. The bins are colored black.
bin.outline.alpha	a number. Alpha value of the bin outline.
highlight.colname	a character. If you desire to color certain points (e.g. significant variants) rather than color by chromosome, you can specify the category in this column, and provide the color mapping in highlight.col. Ignored if NULL.
highlight.counts	a logical. If logical, the bins are colored based on the number of points in each block.
bin.palette	a character. Palette to color the bins. Only palettes supported by the package paletteer are supported. More in details.
bin.alpha	a number. Alpha value of the bins.
palette.direction	a number. Direction of the palette. 1 for increasing and -1 for decreasing.
nonsignif.default	a character. Default color for bins that are not significant.
show.legend	a logical. Show legend if bins are colored based on a variable.

`legend.title` a character. Title of the legend.

`background.col` a character. Color of the background panels. Set to `NULL` for no color.

`background.alpha`
a number. Alpha value of the background panels.

`plot.title` a character. Plot title

`plot.subtitle` a character. Plot subtitle

`plot.width` a numeric. Plot width in inches. Corresponds to `width` argument in `ggsave` function.

`plot.height` a numeric. Plot height in inches. Corresponds to `height` argument in `ggsave` function.

`plot.scale` a numeric. Plot scale. Corresponds to `scale` argument in `ggsave` function.

`bins.x` an integer. number of blocks to horizontally span the longest chromosome

`bins.y` an integer. number of blocks to vertically span the plot

`chr.gap.scaling`
a numeric. scaling factor for gap between chromosome if you desire to change it if x is an `MPdata` object, then the gap will scale relative to the gap in the object.

`signif` a numeric vector. Significant p-value thresholds to be drawn for manhattan plot. At least one value should be provided. Default value is c(5e-08, 1e-5). If `signif` is not `NULL` and `x` is an `MPdata` object, `signif` argument overrides the value inside `MPdata`.

`pval.colname` a character. Column name of `x` containing p.value.

`chr.colname` a character. Column name of `x` containing chromosome.

`pos.colname` a character. Column name of `x` containing position.

`chr.order` a character vector. Order of chromosomes presented in manhattan plot.

`signif.col` a character vector of equal length as `signif`. It contains colors for the lines drawn at `signif`. If `NULL`, the smallest value is colored black while others are grey. If `x` is an `MPdata` object, behaves similarly to `signif`.

`preserve.position`
a logical. If `TRUE`, the width of each chromosome reflect the number of variants and the position of each variant is correctly scaled? If `FALSE`, the width of each chromosome is equal and the variants are equally spaced.

`pval.log.transform`
a logical. If `TRUE`, the p-value will be transformed to -log10(p-value).

`summarise.expression.list`
a list of formulas to summarise data for each bin. Check details for more information.

Details

Similar to `manhattan_plot`, this function accepts summary statistics from GWAS and plots manhattan plot. The difference is that the variants are binned. The number of blocks can be controlled by `bins.x` and `bins.y`. The blocks can be colored based on a column in the data frame.

Palette for coloring the bins can be chosen from the package paletteer. Only palettes available in paletteer are supported. Furthermore, what palette you can use depends on what kind of variable you are using to fill the bins. Use discrete palette for categorical variable and continuous palette for continuous variable.

`binned_manhattan_preprocess` *Preprocess GWAS Result for Binned Manhattan Plot*

Description

Preprocess a result from Genome Wide Association Study before creating a binned manhattan plot. Works similar to `manhattan_data_preprocess`. Returns a `MPdataBinned` object. It can be created using a `data.frame` or a `MPdata` object. Go to details to read how to use `summarise.expression.list`.

Usage

```
binned_manhattan_preprocess(x, ...)

## Default S3 method:
binned_manhattan_preprocess(x, ...)

## S3 method for class 'MPdata'
binned_manhattan_preprocess(
  x,
  bins.x = 10,
  bins.y = 100,
  chr.gap.scaling = 0.4,
  summarise.expression.list = NULL,
  show.message = TRUE,
  ...
)

## S3 method for class 'data.frame'
binned_manhattan_preprocess(
  x,
  bins.x = 10,
  bins.y = 100,
  chr.gap.scaling = 0.4,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.colname = "chr",
  pos.colname = "pos",
  chr.order = NULL,
  signif.col = NULL,
  preserve.position = TRUE,
  pval.log.transform = TRUE,
```

```
  summarise.expression.list = NULL,
  ...
)

## S4 method for signature 'GRanges'
binned_manhattan_preprocess(
  x,
  bins.x = 10,
  bins.y = 100,
  chr.gap.scaling = 0.4,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.order = NULL,
  signif.col = NULL,
  preserve.position = TRUE,
  pval.log.transform = TRUE,
  summarise.expression.list = NULL,
  ...
)
```

Arguments

x	a data.frame or any other extension of a data frame. It can also be a MPdata object.
...	Ignored
bins.x	an integer. number of blocks to horizontally span the longest chromosome
bins.y	an integer. number of blocks to vertically span the plot
chr.gap.scaling	a number. scaling factor for the gap between chromosomes
summarise.expression.list	a list of formulas to summarise data for each bin. Check details for more information.
show.message	a logical. Show warning if MPdata directly used. Set to FALSE to suppress warning.
signif	a numeric vector. Significant p-value thresholds to be drawn for manhattan plot. At least one value should be provided. Default value is c(5e-08, 1e-5)
pval.colname	a character. Column name of x containing p.value.
chr.colname	a character. Column name of x containing chromosome.
pos.colname	a character. Column name of x containing position.
chr.order	a character vector. Order of chromosomes presented in manhattan plot.
signif.col	a character vector of equal length as signif. It contains colors for the lines drawn at signif. If NULL, the smallest value is colored black while others are grey.
preserve.position	a logical. If TRUE, the width of each chromosome reflect the number of variants and the position of each variant is correctly scaled? If FALSE, the width of each chromosome is equal and the variants are equally spaced.

pval.log.transform
: a logical. If TRUE, the p-value will be transformed to -log10(p-value).

Details

If x is a data frame or something alike, then it creates a MPdata object first and then builds MPdataBinned S3 object.

x can also be a MPdata object. Be sure to check if thin has been applied because this can affect what's being aggregated such as number of variables in each bin.

Positions of each point relative to the plot are first calculated via manhattan_data_preprocess. Then the data is binned into blocks. bins.x indicates number of blocks allocated to the chromsome with the widest width. The number of blocks for other chromosomes is proportional to the widest chromosome. bins.y indicates the number of blocks allocated to the y-axis. The number may be slightly adjusted to have the block height end exactly at the significance threshold.

Since points are aggregated into bins, users have the choice to freely specify expressions to summarise the data for each bin through summarise.expression.list argument. This argument takes a list of two-sided formulas, where the left side is the name of the new column and the right side is the expression to calculate the column. This expression is then passed to summarise. For example, to calculate the mean, min, max of a column named beta in each bin, summarise.expression.list arument would be

```
# inside binned_manhattan_preprocess function
summarise.expression.list = list(
  mean_beta ~ mean(beta),
  min_beta ~ min(beta),
  max_beta ~ max(beta)
)
```

Value

a MPdataBinned object. This object contains necessary components for creating a binned manhattan plot.

Examples

```
gwasdat <- data.frame(
  "chromosome" = rep(1:5, each = 1500),
  "position" = c(replicate(5, sample(1:15000, 30))),
  "pvalue" = rbeta(7500, 1, 1)^5,
  "beta" = rnorm(7500)
)

tmp <- binned_manhattan_preprocess(
  gwasdat, pval.colname = "pvalue", chr.colname = "chromosome",
  pos.colname = "position", chr.order = as.character(1:5),
  bins.x = 10, bins.y = 50,
  summarise.expression.list = list(
    mean_beta ~ mean(beta, na.rm = TRUE),
    max_abs_beta ~ max(abs(beta), na.rm = TRUE)
  )
```

```
)

print(tmp)
```

`calc_new_pos` *Calculate new x-position of each point*

Description

Calculate the actual x-positions of each point used for the manhattan plot. `MPdata` object contains the unscaled positions that has not been positioned according to the relative position and width of each chromosome.

Usage

```
calc_new_pos(mpdata)
```

Arguments

`mpdata`	an `MPdata` object.

Details

This is used calculate the actual positions used for the inside `manhattan_plot` function. It was designed this way should the scaling and relative positioning of each chromosome be changed (e.g. gap between the)

Value

a `numeric` vector containing the scaled x-positions.

Examples

```
gwasdat <- data.frame(
  "chromosome" = rep(1:5, each = 30),
  "position" = c(replicate(5, sample(1:300, 30))),
  "pvalue" = rbeta(150, 1, 1)^5
)

mpdata <- manhattan_data_preprocess(
  gwasdat, pval.colname = "pvalue", chr.colname = "chromosome", pos.colname = "position",
  chr.order = as.character(1:5)
)

calc_new_pos(mpdata)
```

default_gds_path *Path to Default GDS File*

Description

Find path to the default gds file.

Usage

```
default_gds_path()
```

Value

A character vector.

Examples

```
default_gds_path()
```

gds_annotate *Annotation with GDS File*

Description

Retrieve variant annotation stored in a GDS file with chromosome location or rs.id.

Usage

```
gds_annotate(
  x,
  gdsfile = NULL,
  annot.method = "position",
  chr = NULL,
  pos = NULL,
  ref = NULL,
  alt = NULL,
  rs.id = NULL,
  concat_char = "/",
  verbose = TRUE,
  annotation_names = c("annotation/info/symbol", "annotation/info/consequence",
    "annotation/info/LoF")
)
```

Arguments

x	a data.frame object to be annotated.
gdsfile	a character for GDS filename. If NULL, the default GDS file included with the package is used.
annot.method	a method for searching variants. "position" requires chr, pos, ref, and alt. "rs.id" requires rs.id.
chr, pos, ref, alt, rs.id	column names of x that contain chromosome, position, reference allele, alternate allele, and rs.id, respectively.
concat_char	a character used to separate multiple annotations returned from the gds file.
verbose	output messages.
annotation_names	a character vector of nodes of the gdsfile that are to be extracted.

Value

A character vector the length of nrow(x) if concat_char is a character. A data frame with nrow(x) rows and length(annotation_names) if concat_char is null.

Examples

```
vardata <- data.frame(
  chr = c(11,20,14),
  pos = c(12261002, 10033792, 23875025),
  ref = c("G", "G", "CG"),
  alt = c("A", "A", "C")
)

annotations <- gds_annotate(
  x = vardata, annot.method = "position",
  chr = "chr", pos = "pos", ref = "ref", alt = "alt"
)

print(annotations)
```

---

ggmanh *ggmanh: A package for visualization of GWAS results.*

Description

ggmanh provides flexible tools for visualizing GWAS result for downstream analysis.

Details

Manhattan plot is commonly used to display significant Single Nucleotide Polymorphisms (SNPs) in Genome Wide Association Study (GWAS) This package comes with features useful for manhattan plot creation, including annotation with `ggrepel`, truncating data for faster plot generation, and manual rescaling of the y-axis. The manhattan plot is generated in two steps: data preprocessing and plotting. This allows the user to iteratively customize the plot without having the process the GWAS summary data over and over again. Currently, `data.frame` and `GRanges` from `GenomicRanges` are supported.

A vignette detailing the usage of the package is accessible by `vignette("ggmanh")`

Author(s)

Maintainer: John Lee <swannyy.stat@gmail.com>

Authors:

- John Lee <john.lee@abbvie.com> (AbbVie)

Other contributors:

- Xiuwen Zheng <xiuwen.zheng@abbvie.com> [contributor, data contributor]

---

ggmanh_annotation_gds *gnomAD Variant Annotation in SeqArray Format*

Description

`ggmanh` provides a GDS file whose path is accessible by `default_gds_path`. The original annotation file is from the gnomAD browser v2.1.1 release, available in this link: https://gnomad.broadinstitute.org/downloads. This gds file contains variants in the exome with the global minor allele frequency ≥ 0.0002, and has been manually curated to fit the file size requirement for R Bioconductor packages.

Format

A GDS file with 1015430 variants with chromosome, position, allele, gene symbol, Ensembl VEP Consequence, and predicted LoF.

manhattan_data_preprocess *Preprocess GWAS Result*

Description

Preprocesses a result from Genome Wide Association Study before making a manhattan plot. It accepts a `data.frame`, which at bare minimum should contain a chromosome, position, and p-value. Additional options, such as chromosome color, label column names, and colors for specific variants, are provided here.

Usage

```
manhattan_data_preprocess(x, ...)

## Default S3 method:
manhattan_data_preprocess(x, ...)

## S3 method for class 'data.frame'
manhattan_data_preprocess(
  x,
  chromosome = NULL,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.colname = "chr",
  pos.colname = "pos",
  highlight.colname = NULL,
  chr.order = NULL,
  signif.col = NULL,
  chr.col = NULL,
  highlight.col = NULL,
  preserve.position = FALSE,
  thin = NULL,
  thin.n = 1000,
  thin.bins = 200,
  pval.log.transform = TRUE,
  chr.gap.scaling = 1,
  ...
)

## S4 method for signature 'GRanges'
manhattan_data_preprocess(
  x,
  chromosome = NULL,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  highlight.colname = NULL,
```

```
  chr.order = NULL,
  signif.col = NULL,
  chr.col = NULL,
  highlight.col = NULL,
  preserve.position = FALSE,
  thin = NULL,
  thin.n = 100,
  thin.bins = 200,
  pval.log.transform = TRUE,
  chr.gap.scaling = 1,
  ...
)
```

Arguments

x	a data frame or any other extension of data frame (e.g. a tibble). At bare minimum, it should contain chromosome, position, and p-value.
...	Additional arguments for manhattan_data_preprocess.
chromosome	a character. This is supplied if a manhattan plot of a single chromosome is desired. If NULL, then all the chromosomes in the data will be plotted.
signif	a numeric vector. Significant p-value thresholds to be drawn for manhattan plot. At least one value should be provided. Default value is c(5e-08, 1e-5)
pval.colname	a character. Column name of x containing p.value.
chr.colname	a character. Column name of x containing chromosome.
pos.colname	a character. Column name of x containing position.
highlight.colname	a character. If you desire to color certain points (e.g. significant variants) rather than color by chromosome, you can specify the category in this column, and provide the color mapping in highlight.col. Ignored if NULL.
chr.order	a character vector. Order of chromosomes presented in manhattan plot.
signif.col	a character vector of equal length as signif. It contains colors for the lines drawn at signif. If NULL, the smallest value is colored black while others are grey.
chr.col	a character vector of equal length as chr.order. It contains colors for the chromosomes. Name of the vector should match chr.order. If NULL, default colors are applied using RColorBrewer.
highlight.col	a character vector. It contains color mapping for the values from highlight.colname.
preserve.position	a logical. If TRUE, the width of each chromosome reflect the number of variants and the position of each variant is correctly scaled? If FALSE, the width of each chromosome is equal and the variants are equally spaced.
thin	a logical. If TRUE, thinPoints will be applied. Defaults to TRUE if chromosome is NULL. Defaults to FALSE if chromosome is supplied.
thin.n	an integer. Number of max points per horizontal partitions of the plot. Defaults to 1000.

`thin.bins` an integer. Number of bins to partition the data. Defaults to 200.

`pval.log.transform`
a logical. If `TRUE`, the p-value will be transformed to -log10(p-value).

`chr.gap.scaling`
scaling factor for gap between chromosome if you desire to change it. This can also be set in `manhattan_plot`

Details

`manhattan_data_preprocess` gathers information needed to plot a manhattan plot and organizes the information as `MPdata` S3 object.

New positions for each points are calculated, and stored in the data.frame as `"new_pos"`. By default, all chromosomes will have the same width, with each point being equally spaced. This behavior is changed when `preserve.position = TRUE`. The width of each chromosome will scale to the number of points and the points will reflect the original positions.

`chr.col` and `highlight.col`, maps the data values to colors. If they are an unnamed vector, then the function will try its best to match the values of `chr.colname` or `highlight.colname` to the colors. If they are a named vector, then they are expected to map all values to a color. If `highlight.colname` is supplied, then `chr.col` is ignored.

While feeding a `data.frame` directly into `manhattan_plot` does preprocessing & plotting in one step. If you plan on making multiple plots with different graphic options, you have the choice to preprocess separately and then generate plots.

Value

a MPdata object. This object contains all the necessary components for constructing a manhattan plot.

Examples

```
gwasdat <- data.frame(
  "chromosome" = rep(1:5, each = 30),
  "position" = c(replicate(5, sample(1:300, 30))),
  "pvalue" = rbeta(150, 1, 1)^5
)

  manhattan_data_preprocess(
  gwasdat, pval.colname = "pvalue", chr.colname = "chromosome", pos.colname = "position",
  chr.order = as.character(1:5)
)
```

manhattan_plot *Manhattan Plotting*

Description

A generic function for manhattan plot.

Usage

```
manhattan_plot(x, ...)

manhattan_plot.default(x, ...)

## S3 method for class 'data.frame'
manhattan_plot(
  x,
  chromosome = NULL,
  outfn = NULL,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  chr.colname = "chr",
  pos.colname = "pos",
  label.colname = NULL,
  highlight.colname = NULL,
  chr.order = NULL,
  signif.col = NULL,
  chr.col = NULL,
  highlight.col = NULL,
  rescale = TRUE,
  rescale.ratio.threshold = 5,
  signif.rel.pos = 0.2,
  chr.gap.scaling = 1,
  color.by.highlight = FALSE,
  preserve.position = FALSE,
  thin = NULL,
  thin.n = 1000,
  thin.bins = 200,
  pval.log.transform = TRUE,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  point.size = 0.75,
  label.font.size = 2,
  max.overlaps = 20,
  x.label = "Chromosome",
```

```
  y.label = expression(-log[10](p)),
  ...
)

## S3 method for class 'MPdata'
manhattan_plot(
  x,
  chromosome = NULL,
  outfn = NULL,
  signif = NULL,
  signif.col = NULL,
  rescale = TRUE,
  rescale.ratio.threshold = 5,
  signif.rel.pos = 0.2,
  chr.gap.scaling = NULL,
  color.by.highlight = FALSE,
  label.colname = NULL,
  x.label = "Chromosome",
  y.label = expression(-log[10](p)),
  point.size = 0.75,
  label.font.size = 2,
  max.overlaps = 20,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  ...
)

## S4 method for signature 'GRanges'
manhattan_plot(
  x,
  chromosome = NULL,
  outfn = NULL,
  signif = c(5e-08, 1e-05),
  pval.colname = "pval",
  label.colname = NULL,
  highlight.colname = NULL,
  chr.order = NULL,
  signif.col = NULL,
  chr.col = NULL,
  highlight.col = NULL,
  rescale = TRUE,
  rescale.ratio.threshold = 5,
  signif.rel.pos = 0.2,
  chr.gap.scaling = 1,
  color.by.highlight = FALSE,
```

```
  preserve.position = FALSE,
  thin = NULL,
  thin.n = 1000,
  thin.bins = 200,
  pval.log.transform = TRUE,
  plot.title = ggplot2::waiver(),
  plot.subtitle = ggplot2::waiver(),
  plot.width = 10,
  plot.height = 5,
  plot.scale = 1,
  point.size = 0.75,
  label.font.size = 2,
  max.overlaps = 20,
  x.label = "Chromosome",
  y.label = expression(-log[10](p)),
  ...
)
```

Arguments

x	a data.frame, an extension of data.frame object (e.g. tibble), or an MPdata object.
...	additional arguments to be passed onto geom_label_repel
chromosome	a character. This is supplied if a manhattan plot of a single chromosome is desired. If NULL, then all the chromosomes in the data will be plotted.
outfn	a character. File name to save the Manhattan Plot. If outfn is supplied (i.e. !is.null(outfn)), then the plot is not drawn in the graphics window.
signif	a numeric vector. Significant p-value thresholds to be drawn for manhattan plot. At least one value should be provided. Default value is c(5e-08, 1e-5). If signif is not NULL and x is an MPdata object, signif argument overrides the value inside MPdata.
pval.colname	a character. Column name of x containing p.value.
chr.colname	a character. Column name of x containing chromosome.
pos.colname	a character. Column name of x containing position.
label.colname	a character. Name of the column in MPdata$data to be used for labeling.
highlight.colname	a character. If you desire to color certain points (e.g. significant variants) rather than color by chromosome, you can specify the category in this column, and provide the color mapping in highlight.col. Ignored if NULL.
chr.order	a character vector. Order of chromosomes presented in manhattan plot.
signif.col	a character vector of equal length as signif. It contains colors for the lines drawn at signif. If NULL, the smallest value is colored black while others are grey. If x is an MPdata object, behaves similarly to signif.
chr.col	a character vector of equal length as chr.order. It contains colors for the chromosomes. Name of the vector should match chr.order. If NULL, default colors are applied using RColorBrewer.

highlight.col a character vector. It contains color mapping for the values from highlight.colname.

rescale a logical. If TRUE, the plot will rescale itself depending on the data. More on this in details.

rescale.ratio.threshold
a numeric. Threshold of that triggers the rescale.

signif.rel.pos a numeric between 0.1 and 0.9. If the plot is rescaled, where should the significance threshold be positioned?

chr.gap.scaling
a numeric. scaling factor for gap between chromosome if you desire to change it if x is an MPdata object, then the gap will scale relative to the gap in the object.

color.by.highlight
a logical. Should the points be colored based on a highlight column?

preserve.position
a logical. If TRUE, the width of each chromosome reflect the number of variants and the position of each variant is correctly scaled? If FALSE, the width of each chromosome is equal and the variants are equally spaced.

thin a logical. If TRUE, thinPoints will be applied. Defaults to TRUE if chromosome is NULL. Defaults to FALSE if chromosome is supplied.

thin.n an integer. Number of max points per horizontal partitions of the plot. Defaults to 1000.

thin.bins an integer. Number of bins to partition the data. Defaults to 200.

pval.log.transform
a logical. If TRUE, the p-value will be transformed to -log10(p-value).

plot.title a character. Plot title

plot.subtitle a character. Plot subtitle

plot.width a numeric. Plot width in inches. Corresponds to width argument in ggsave function.

plot.height a numeric. Plot height in inches. Corresponds to height argument in ggsave function.

plot.scale a numeric. Plot scale. Corresponds to scale argument in ggsave function.

point.size a numeric. Size of the points.

label.font.size
a numeric. Size of the labels.

max.overlaps an integer. Exclude text labels that overlaps too many things.

x.label a character. x-axis label

y.label a character. y-axis label

Details

This generic function accepts a result of a GWAS in the form of data.frame or a MPdata object produced by manhattan_data_preprocess. The function will throw an error if another type of object is passed.

Having `rescale = TRUE` is useful when there are points with very high -log10(p.value). In this case, the function attempts to split the plot into two different scales, with the split happening near the strictest significance threshold. More precisely, the plot is rescaled when

$$-log10(pvalue)/(strictestsignificancethreshold) \geq rescale.ratio.threshold$$

If you wish to add annotation to the points, provide the name of the column to `label.colname`. The labels are added with `ggrepel`.

Be careful though: if the annotation column contains a large number of variants, then the plotting could take a long time, and the labels will clutter up the plot. For those points with no annotation, you have the choice to set them as `NA` or `""`.

Value

gg object if `is.null(outfn)`, `NULL` if `!is.null(outf)`

Examples

```
gwasdat <- data.frame(
  "chromosome" = rep(1:5, each = 30),
  "position" = c(replicate(5, sample(1:300, 30))),
  "pvalue" = rbeta(150, 1, 1)^5
)

manhattan_plot(
  gwasdat, pval.colname = "pvalue", chr.colname = "chromosome", pos.colname = "position",
  chr.order = as.character(1:5)
)

mpdata <- manhattan_data_preprocess(
  gwasdat, pval.colname = "pvalue", chr.colname = "chromosome", pos.colname = "position",
  chr.order = as.character(1:5)
)

manhattan_plot(mpdata)
```

---

qqunif *Plot Quantile-Quantile Plot of p-values against uniform distribution.*

Description

Plot Quantile-Quantile Plot of p-values against uniform distribution.

Usage

```
qqunif(
  x,
  outfn = NULL,
  conf.int = 0.95,
  plot.width = 5,
  plot.height = 5,
  thin = TRUE,
  thin.n = 500,
  zero.pval = "replace"
)
```

Arguments

x	a numeric vector of p-values. All values should be between 0 and 1.
outfn	a character. File name to save the QQ Plot. If `outfn` is supplied (i.e. `!is.null(outfn)`), then the plot is not drawn in the graphics window.
conf.int	a numeric between 0 and 1. Confidence band to draw around reference line. Set to `NA` to leave it out.
plot.width	a numeric. Plot width in inches.
plot.height	a numeric. Plot height in inches.
thin	a logical. Reduce number of data points when they are cluttered?
thin.n	an integer. Number of max points per horizontal partitions of the plot. Defaults to 500.
zero.pval	a character. Determine how to treat 0 pvals. "replace" will replace the p-value of zero with the non-zero minimum. "remove" will remove the p-value of zero.

Value

a ggplot object

Examples

```
x <- rbeta(1000, 1, 1)
qqunif(x)
```

---

thinPoints *Thin Data Points*

Description

Reduce the number of cluttered data points.

Usage

```
thinPoints(dat, value, n = 1000, nbins = 200, groupBy = NULL)
```

Arguments

dat	a data frame
value	column name of dat to be used for partitioning (see details)
n	number of points to sample for each partition
nbins	number of partitions
groupBy	column name of dat to group by before partitioning (e.g. chromosome)

Details

The result of Genome Wide Association Study can be very large, with the majority of points being being clustered below significance threshold. This unnecessarily increases the time to plot while making almost no difference. This function reduces the number of points by partitioning the points by a numeric column value into nbins and sampling n points.

Value

a data.frame

Examples

```
dat <- data.frame(
   A1 = c(1:20, 20, 20),
   A2 = c(rep(1, 12), rep(1,5), rep(20, 3), 20, 20) ,
   B = rep(c("a", "b", "c", "d"), times = c(5, 7, 8, 2))
)
# partition "A1" into 2 bins and then sample 6 data points
thinPoints(dat, value = "A1", n = 6, nbins = 2)
# partition "A2" into 2 bins and then sample 6 data points
thinPoints(dat, value = "A2", n = 6, nbins = 2)
# group by "B", partition "A2" into 2 bins and then sample 3 data points
thinPoints(dat, value = "A2", n = 3, nbins = 2, groupBy = "B")
```

Index

∗ **internal**
ggmanh-package, 2

binned_manhattan_plot, 3
binned_manhattan_plot,GRanges-method *(*binned_manhattan_plot*)*, 3
binned_manhattan_plot.data.frame *(*binned_manhattan_plot*)*, 3
binned_manhattan_plot.default *(*binned_manhattan_plot*)*, 3
binned_manhattan_plot.MPdataBinned *(*binned_manhattan_plot*)*, 3
binned_manhattan_preprocess, 7
binned_manhattan_preprocess,GRanges-method *(*binned_manhattan_preprocess*)*, 7
binned_manhattan_preprocess.data.frame *(*binned_manhattan_preprocess*)*, 7
binned_manhattan_preprocess.default *(*binned_manhattan_preprocess*)*, 7
binned_manhattan_preprocess.MPdata *(*binned_manhattan_preprocess*)*, 7

calc_new_pos, 10

default_gds_path, 11

gds_annotate, 11
ggmanh, 12
ggmanh-package, 2
ggmanh_annotation_gds, 13
ggrepel, *13*, *21*

manhattan_data_preprocess, *7*, *9*, 14
manhattan_data_preprocess,GRanges-method *(*manhattan_data_preprocess*)*, 14
manhattan_data_preprocess.data.frame *(*manhattan_data_preprocess*)*, 14
manhattan_data_preprocess.default *(*manhattan_data_preprocess*)*, 14
manhattan_plot, 17
manhattan_plot,GRanges-method *(*manhattan_plot*)*, 17
manhattan_plot.data.frame *(*manhattan_plot*)*, 17
manhattan_plot.default *(*manhattan_plot*)*, 17
manhattan_plot.MPdata *(*manhattan_plot*)*, 17

qqunif, 21

summarise, *9*

thinPoints, 22