Package: ggwas (via r-universe)

November 27, 2026

Title Modern ggplot2 Visualizations for Genome-Wide Association Studies

Version 0.99.13

Description Create publication-ready visualizations for genome-wide association studies. Provides 21 plot types including Manhattan, value Manhattan, QQ, Miami, locus zoom, PheWAS, colocalization, fine-mapping, genetic correlation, SNP density, and density-signal comparison plots. Reads PLINK, REGENIE, GCTA, and GEMMA formats natively and accepts variants as 'GRanges'. Gene models, annotation regions and chromosome lengths come from Bioconductor objects ('GRanges', 'TxDb', 'EnsDb', 'Seqinfo'). All plot functions return 'ggplot2' objects for full customization. Smart downsampling handles datasets with 10 million+ variants.

License MIT + file LICENSE

URL https://github.com/bczech/ggwas, https://bczech.github.io/ggwas/

BugReports https://github.com/bczech/ggwas/issues

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

Depends R (>= 4.5.0)

Imports cli, data.table (>= 1.14.0), ggplot2 (>= 3.5.0), ggrepel (>= 0.9.0), GenomeInfoDb, grDevices, grid, gridExtra, methods, patchwork (>= 1.2.0), RColorBrewer, rlang (>= 1.0.0), scales, stats, utils, withr, GenomicRanges, IRanges, S4Vectors

Suggests AnnotationDbi, BiocStyle, EnsDb.Hsapiens.v86, ensembldb, GenomicFeatures, knitr, org.Hs.eg.db, rmarkdown, rtracklayer, testthat (>= 3.0.0), TxDb.Hsapiens.UCSC.hg38.knownGene

VignetteBuilder knitr

BiocType Software

biocViews Visualization, GenomeWideAssociation, Annotation, DataImport, Software

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libssl-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-11-24 17:27:40 UTC

RemoteUrl https://github.com/bioc/ggwas

RemoteRef HEAD

RemoteSha 79d96dc6d13405c463fbdd6a8f23d930c9b81aad

Contents

annotate_genes . . . 3
annotate_positions . . . 4
architecture_plot . . . 5
as_granges . . . 6
as_gwas_data . . . 7
calc_lambda . . . 8
circular_manhattan . . . 9
coloc_plot . . . 11
density_signal_plot . . . 12
effect_compare_plot . . . 13
enrichment_manhattan . . . 15
example_gwas . . . 17
filter_region . . . 18
finemapping_plot . . . 18
forest_plot . . . 20
gene_track . . . 21
genetic_correlation . . . 23
get_loci . . . 24
gwas_palette . . . 25
gwas_palettes . . . 25
gwas_preset . . . 26
gwas_summary . . . 26
highlight_regions . . . 28
journal_themes . . . 29
locus_plot . . . 30
maf_filter . . . 32
manhattan_genes . . . 32
manhattan_plot . . . 33
merge_gwas . . . 36
miami_plot . . . 37
multitrait_manhattan . . . 38
phewas_plot . . . 40

pvalue_heatmap 41
qq_plot 42
read_gcta_mlma 44
read_gemma 45
read_gwas_table 45
read_plink_assoc 46
read_plink_linear 47
read_plink_logistic 47
read_regenie 48
scale_color_chromosome 49
scale_color_gwas 50
set_sex_chr_map 50
snp_density 51
theme_gwas 53
top_hits 54
trumpet_plot 55
validate_gwas_data 56
value_manhattan 57
volcano_plot 59

Index **61**

`annotate_genes` *Annotate peaks with nearest gene names*

Description

Maps significant SNPs to their nearest gene with `GenomicRanges::distanceToNearest()`.

Usage

```
annotate_genes(
  data,
  genes,
  p_threshold = 5e-08,
  max_distance = 5e+05,
  top_n = NULL,
  clump_window = 1e+06
)
```

Arguments

`data`	A `gwas_data` object, data.frame or `GRanges` of variants.
`genes`	Gene models as a GenomicRanges::GRanges, `TxDb` or `EnsDb` object. Gene names are taken as described in `gene_track()`.
`p_threshold`	Only annotate SNPs with p < this value.

`max_distance`	Maximum distance (bp) between a SNP and the nearest gene. The distance is 0 for a SNP inside a gene and otherwise the gap to the closest gene end. Default 500kb.
`top_n`	Annotate only the top N most significant SNPs.
`clump_window`	Clumping window in bp. Within each window, only the most significant SNP is annotated. Default 1Mb.

Value

The input as a `gwas_data` object with added `nearest_gene` and `gene_distance` columns.

Examples

```
data(example_gwas)
genes <- GenomicRanges::GRanges(
  c("chr1:1000000-1500000", "chr2:50000000-51000000",
    "chr5:80000000-82000000"),
  gene_name = c("GENE_A", "GENE_B", "GENE_C")
)
annotated <- annotate_genes(example_gwas, genes, top_n = 10)
head(annotated[!is.na(annotated$nearest_gene), ])
```

`annotate_positions` *Map gene IDs or symbols to genome coordinates*

Description

Look up genomic positions for a set of gene identifiers in a gene annotation. This is the inverse of `annotate_genes()`, which maps variant positions to the nearest gene; here you start from gene identifiers and get their coordinates, for example to feed into `value_manhattan()`.

Usage

```
annotate_positions(ids, annotation, id_type = c("gene_id", "symbol"))
```

Arguments

`ids`	Character vector of gene identifiers (Ensembl gene IDs such as `"ENSG00000141510"`, Entrez IDs, or gene symbols such as `"TP53"`).
`annotation`	Gene models as a GenomicRanges::GRanges object, or a `TxDb` (**GenomicFeatures**) or `EnsDb` (**ensembldb**) object. Gene IDs are read from the `gene_id` metadata column (or `names()`), symbols from `gene_name`, `symbol` or `gene`. GTF and GFF3 files can be read with `rtracklayer::import()`; keep the `type == "gene"` rows.
`id_type`	Whether `ids` are gene IDs (`"gene_id"`, default) or gene symbols (`"symbol"`).

Value

A GenomicRanges::GRanges with one range per matched identifier and metadata columns `id` (the input identifier), `gene` (symbol) and `gene_id`. Identifiers that cannot be mapped are dropped.

See Also

`value_manhattan()`, `annotate_genes()`

Examples

```
genes <- GenomicRanges::GRanges(
  c("chr17:7661779-7687538:-", "chr17:43044295-43170245:-"),
  gene_id = c("ENSG00000141510", "ENSG00000012048"),
  gene_name = c("TP53", "BRCA1")
)
annotate_positions(c("ENSG00000141510", "ENSG00000012048"), genes)
annotate_positions(c("TP53", "NOT_A_GENE"), genes, id_type = "symbol")
```

`architecture_plot` *Genetic architecture plot*

Description

Visualize the relationship between minor allele frequency and effect size, revealing whether a trait's genetic architecture is polygenic (many small effects) or oligogenic (few large effects). Points are colored by significance.

Usage

```
architecture_plot(
  data,
  beta = NULL,
  p = NULL,
  af = NULL,
  p_threshold = 5e-08,
  colors = c(significant = "#E64B35", nonsignificant = "#BDC3C7"),
  point_size = 1,
  alpha = 0.5,
  log_maf = FALSE,
  show_density = FALSE,
  label_top_n = NULL,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame with P, BETA, and AF columns.
`beta`, `p`, `af`	Column name overrides.
`p_threshold`	P-value threshold for coloring significant variants.
`colors`	Named vector with "significant" and "nonsignificant" colors.
`point_size`	Point size.
`alpha`	Transparency.
`log_maf`	If TRUE, use log10(MAF) on x-axis.
`show_density`	If TRUE, add marginal density curves.
`label_top_n`	Label the top N variants by significance.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas)

# Basic architecture plot
architecture_plot(example_gwas)

# Label top hits, lower threshold for demo
architecture_plot(example_gwas, p_threshold = 0.001, label_top_n = 5)

# Log-scale MAF axis
architecture_plot(example_gwas, p_threshold = 0.001, log_maf = TRUE)
```

`as_granges` *Convert GWAS results to a GRanges object*

Description

Turn a `gwas_data` object or data.frame into a Bioconductor GRanges object, with association statistics stored as metadata columns, for interoperability with the Bioconductor ecosystem (e.g. VariantAnnotation, SummarizedExperiment, coloc).

Usage

```
as_granges(x, ...)
```

Arguments

`x`	A `gwas_data` object or a data.frame accepted by `as_gwas_data()`.
`...`	Column overrides passed to `as_gwas_data()` when `x` is a plain data.frame.

Value

A GRanges object, one width-1 range per variant, named by SNP when available.

Examples

```
data(example_gwas)
gr <- as_granges(example_gwas)
gr

# Round-trip back to gwas_data
as_gwas_data(gr)
```

`as_gwas_data` *Create a gwas_data object*

Description

Convert a data.frame to the standardized gwas_data format used by all ggwas plotting functions.

Usage

```
as_gwas_data(
  x,
  chr = NULL,
  bp = NULL,
  snp = NULL,
  p = NULL,
  beta = NULL,
  se = NULL,
  a1 = NULL,
  a2 = NULL,
  af = NULL,
  n = NULL,
  info = NULL,
  log_p = FALSE
)
```

Arguments

`x`	A data.frame, tibble, data.table, or a Bioconductor `GRanges` object (variant positions are taken from the ranges and association statistics from the metadata columns).
`chr`, `bp`, `snp`, `p`, `beta`, `se`, `a1`, `a2`, `af`, `n`, `info`	Column names to use. If NULL, auto-detection is attempted.
`log_p`	If TRUE, the p-value column contains -log10(p) values. Use this for p-values smaller than R can store (about 1e-323).

Value

A data.frame with class `gwas_data`. P-values below the double range are stored as `P = 0`; their exact -log10(p) is then kept in a `LOG10P` column, which all plots use. It comes from a -log10(p) column (`log_p = TRUE`), from p-values read as text (e.g. `"1e-400"`, as `data.table::fread()` returns them), or from a `LOG10P` column next to the p-values.

Examples

```
df <- data.frame(CHR = c(1, 1, 2), BP = c(1e6, 2e6, 5e6),
                 P = c(1e-8, 0.5, 0.01), SNP = c("rs1", "rs2", "rs3"))
gd <- as_gwas_data(df)
gd
```

`calc_lambda` *Calculate genomic inflation factor (lambda GC)*

Description

Calculate genomic inflation factor (lambda GC)

Usage

```
calc_lambda(p)
```

Arguments

`p`	Numeric vector of p-values.

Value

Genomic inflation factor (lambda).

Examples

```
data(example_gwas)
calc_lambda(example_gwas$P)
```

`circular_manhattan` *Circular Manhattan plot*

Description

A circos-style Manhattan plot where chromosomes are arranged in a circle. Built entirely with ggplot2, unlike CMplot's base R implementation. Supports multiple rings for multi-trait comparison.

Usage

```
circular_manhattan(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  colors = c("#1A5276", "#76D7C4"),
  point_size = 1.2,
  alpha = 0.85,
  genome_wide = 5e-08,
  threshold_color = "#E74C3C",
  highlight_snps = NULL,
  highlight_color = "#E74C3C",
  highlight_size = 2,
  label_snps = NULL,
  label_top_n = NULL,
  show_chr_labels = TRUE,
  inner_radius = 0.35,
  chr_gap_fraction = 0.012,
  ring_labels = NULL,
  downsample = TRUE,
  downsample_n = 1e+05,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object, data.frame, or named list of gwas_data objects for multi-ring display.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`colors`	Two-color vector for alternating chromosomes, or a palette name from `gwas_palette()`.
`point_size`	Point size.
`alpha`	Point transparency.
`genome_wide`	Genome-wide significance threshold.

`threshold_color`
: Color for threshold ring.

`highlight_snps`
: SNP IDs to highlight.

`highlight_color`
: Color for highlighted SNPs.

`highlight_size`
: Size for highlighted points.

`label_snps` SNP IDs to label.

`label_top_n` Label top N SNPs.

`show_chr_labels`
: Show chromosome labels.

`inner_radius` Inner radius of the circle (proportion, 0-1).

`chr_gap_fraction`
: Gap between chromosomes as fraction of total.

`ring_labels` For multi-ring: labels for each ring.

`downsample` Enable smart downsampling.

`downsample_n` Target points after downsampling.

`title` Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas)

# Single trait
circular_manhattan(example_gwas)

# Nature palette
circular_manhattan(example_gwas, colors = gwas_palette("nature"))

# Multi-ring: two traits
trait2 <- example_gwas
trait2$P <- runif(nrow(trait2))^3
circular_manhattan(list(Trait1 = example_gwas, Trait2 = trait2))
```

`coloc_plot` *Colocalization locus plot*

Description

Two association signals on a shared genomic coordinate axis. This is the standard way to visually assess whether a GWAS signal and an eQTL (or two GWAS traits) share the same causal variant.

Usage

```
coloc_plot(
  gwas1,
  gwas2,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  region = NULL,
  lead_snp = NULL,
  flank = 5e+05,
  ld = NULL,
  ld_colors = c("#2166AC", "#67A9CF", "#78C679", "#F4A582", "#D73027"),
  top_title = "Trait 1",
  bottom_title = "Trait 2",
  highlight_snps = NULL,
  point_size = 2,
  title = NULL
)
```

Arguments

`gwas1`	First dataset (e.g., GWAS). A `gwas_data` or data.frame.
`gwas2`	Second dataset (e.g., eQTL). Same format.
`chr`, `bp`, `p`, `snp`	Column name overrides (applied to both datasets).
`region`	The region to plot: a GenomicRanges::GRanges of length 1, or a string such as `"chr1:1000000-30000000"`. Not needed when `lead_snp` is given.
`lead_snp`	SNP ID to center on (± flank).
`flank`	Flank size in bp around lead_snp.
`ld`	Named numeric vector of LD r^2 values (optional).
`ld_colors`	Colors for LD bins.
`top_title`	Label for the top panel.
`bottom_title`	Label for the bottom panel.
`highlight_snps`	SNPs to highlight in both panels.

point_size	Point size.
title	Overall title.

Value

A patchwork composition of two locus plots.

Examples

```
data(example_gwas)
gwas2 <- example_gwas
gwas2$P <- runif(nrow(gwas2))^2
coloc_plot(example_gwas, gwas2,
           region = "chr1:1000000-30000000",
           top_title = "GWAS", bottom_title = "eQTL")
```

---

density_signal_plot *Density–signal dual-track plot*

Description

Compare SNP genotyping density against association signal strength across the genome. Each chromosome has two rows: the top track shows the number of variants per bin (density), and the bottom track shows the minimum p-value per bin (signal). This helps distinguish genuine association signals from artifacts driven by uneven genotyping or imputation coverage.

Usage

```
density_signal_plot(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  bin_size = 1e+06,
  seqinfo = NULL,
  density_palette = "viridis",
  signal_palette = "magma",
  chromosomes = NULL,
  title = NULL
)
```

Arguments

data	A gwas_data object or data.frame with GWAS results.
chr, bp, p	Column name overrides.
bin_size	Bin size in base pairs (default 1 Mb).

`seqinfo` Optional chromosome lengths as a `Seqinfo` object (**GenomeInfoDb**) or any object that has one, such as a `TxDb`, `EnsDb` or `BSgenome`. If NULL, chromosome lengths are inferred from the data.

`density_palette` Color palette for the density track.

`signal_palette` Color palette for the signal track.

`chromosomes` Optional integer vector of chromosomes to display.

`title` Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")
density_signal_plot(example_gwas, bin_size = 5e6)

# Full chromosome lengths from a Seqinfo. With internet access,
# GenomeInfoDb::Seqinfo(genome = "hg38")[paste0("chr", 1:3)] gives the
# same object; here it is taken offline from a TxDb package.
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
  library(GenomeInfoDb)
  txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene
  si <- seqinfo(txdb)[paste0("chr", 1:3)]
  density_signal_plot(example_gwas, bin_size = 5e6, seqinfo = si,
    chromosomes = 1:3)
}

# Subset to specific chromosomes
density_signal_plot(example_gwas, bin_size = 5e6, chromosomes = seq_len(5))

# Different palettes
density_signal_plot(example_gwas, bin_size = 5e6,
  density_palette = "plasma", signal_palette = "inferno")
```

`effect_compare_plot` *Compare variant effects between two GWAS*

Description

Scatter the per-variant effect sizes of two studies against each other for the variants they share, to assess replication and effect concordance (for example discovery versus replication, or two ancestries). A `y = x` reference line is drawn; points are colored by whether the two effects agree in sign among variants significant in either study.

Usage

```
effect_compare_plot(
  gwas1,
  gwas2,
  snp = NULL,
  beta = NULL,
  se = NULL,
  p = NULL,
  labels = c("Study 1", "Study 2"),
  p_threshold = 5e-08,
  show_ci = TRUE,
  ci = 0.95,
  label_top_n = 10,
  colors = c(concordant = "#2C7FB8", discordant = "#D7301F", ns = "#BDC3C7"),
  point_size = 1.8,
  title = NULL
)
```

Arguments

gwas1, gwas2	gwas_data objects or data.frames. Variants are matched on the SNP column.
snp, beta, se, p	Column name overrides applied to both inputs when they are plain data.frames.
labels	Length-2 character vector of axis labels.
p_threshold	Significance threshold used to classify variants.
show_ci	If TRUE, draw confidence-interval crosses for the significant variants (requires SE columns).
ci	Confidence level for the crosses.
label_top_n	Label the N most significant shared variants.
colors	Named colors for "concordant", "discordant" and "ns".
point_size	Point size.
title	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas)
g2 <- example_gwas
g2$BETA <- g2$BETA + stats::rnorm(nrow(g2), 0, 0.02)
effect_compare_plot(example_gwas, g2,
                    labels = c("Discovery", "Replication"),
                    p_threshold = 1e-3)
```

`enrichment_manhattan` *Enrichment Manhattan plot*

Description

A Manhattan plot with colored overlays for functional genomic annotations. SNPs falling within annotated regions (genes, enhancers, eQTLs, custom categories) are highlighted with distinct colors, connecting GWAS results with functional genomics.

Usage

```
enrichment_manhattan(
  data,
  annotations,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  background_color = "grey75",
  background_alpha = 0.4,
  annotation_colors = NULL,
  annotation_size = 1.5,
  annotation_alpha = 0.9,
  annotation_shape = 16,
  show_legend = TRUE,
  legend_position = "right",
  genome_wide = 5e-08,
  suggestive = 1e-05,
  label_top_n = NULL,
  label_column = "SNP",
  downsample = TRUE,
  downsample_n = 2e+05,
  title = NULL,
  palette = "nature"
)
```

Arguments

`data`	A `gwas_data` object, data.frame or `GRanges` of variants.
`annotations`	Annotation regions as a GenomicRanges::GRanges with an optional `category` metadata column (annotation type), or a named `GRangesList` (or named list of `GRanges`) whose names are used as categories. BED, GFF or bigBed files can be read with `rtracklayer::import()`. A variant falling in several regions takes the category of the first one.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`background_color`	Color for non-annotated SNPs.

`background_alpha`
: Transparency of background points.

`annotation_colors`
: Named vector of colors for each annotation category. If NULL, colors are taken from `palette`.

`annotation_size`
: Point size for annotated SNPs.

`annotation_alpha`
: Transparency of annotated SNPs.

`annotation_shape`
: Point shape for annotated SNPs (default 16, circle).

`show_legend`
: Show legend for annotation categories.

`legend_position`
: Legend position.

`genome_wide`
: Genome-wide significance threshold.

`suggestive`
: Suggestive significance threshold.

`label_top_n`
: Label top N annotated significant SNPs.

`label_column`
: Column to use for labels.

`downsample`
: Enable smart downsampling.

`downsample_n`
: Target number of background points.

`title`
: Plot title.

`palette`
: Color palette for annotations (from `gwas_palette()`).

Value

A ggplot object.

Examples

```
data(example_gwas)
# Annotation regions with a category column
annot <- GenomicRanges::GRanges(
  c("chr1:1000000-5000000", "chr2:50000000-55000000",
    "chr5:100000000-110000000"),
  category = c("Enhancer", "Gene", "eQTL")
)
enrichment_manhattan(example_gwas, annotations = annot)

# The same as a GRangesList, one element per category
annot_list <- GenomicRanges::GRangesList(
  Enhancer = annot[1], Gene = annot[2], eQTL = annot[3]
)
enrichment_manhattan(example_gwas, annotations = annot_list)
```

`example_gwas` *Example GWAS dataset*

Description

A simulated GWAS dataset with 8000 variants across 22 autosomal chromosomes, sized proportionally to real human chromosome lengths. Contains 20 genome-wide significant variants for demonstration.

Usage

```
data(example_gwas)
```

Format

A `gwas_data` data.frame with 8000 rows and 9 columns:

CHR Chromosome (integer, 1-22)

BP Base pair position

SNP Variant identifier (rs ID)

P P-value

BETA Effect size

SE Standard error

A1 Effect allele

A2 Other allele

AF Allele frequency

Value

A `gwas_data` data.frame, loaded into the workspace by `data(example_gwas)`.

Source

Simulated with `data-raw/example_gwas.R` in the package source. Positions are drawn within GRCh38 (hg38) chromosome lengths.

Examples

```
data(example_gwas)
manhattan_plot(example_gwas)
```

filter_region *Filter variants by genomic region*

Description

Subset a GWAS dataset to variants within a specified genomic region.

Usage

```
filter_region(data, chr, start, end)
```

Arguments

data	A gwas_data object or data.frame with CHR and BP columns.
chr	Chromosome (integer or string like "chr6").
start	Start position (bp).
end	End position (bp).

Value

A filtered gwas_data object.

Examples

```
data(example_gwas)
mhc <- filter_region(example_gwas, chr = 6, start = 25e6, end = 34e6)
```

finemapping_plot *Fine-mapping credible set plot*

Description

A locus plot where point size encodes posterior inclusion probability (PIP) from fine-mapping tools like SuSiE or FINEMAP. Credible set membership is shown by color. This is the emerging standard for visualizing fine-mapping results in GWAS publications.

Usage

```
finemapping_plot(
  data,
  pip = "PIP",
  credible_set = "credible_set",
  chr = NULL,
  bp = NULL,
  p = NULL,
```

```
  snp = NULL,
  region = NULL,
  lead_snp = NULL,
  flank = 5e+05,
  pip_min_size = 0.3,
  pip_max_size = 5,
  set_colors = c("#E64B35", "#4DBBD5", "#00A087", "#3C5488", "#F39B7F"),
  nonsig_color = "grey70",
  show_pip_legend = TRUE,
  label_pip_above = 0.5,
  title = NULL
)
```

Arguments

`data`	A data.frame with at minimum CHR, BP, P columns, plus a PIP column (posterior inclusion probability, 0-1).
`pip`	Column name for posterior inclusion probability.
`credible_set`	Column name for credible set assignment (integer). Variants in the same set get the same color.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`region`	The region to plot: a GenomicRanges::GRanges of length 1, or a string such as `"chr1:1000000-50000000"`. Not needed when `lead_snp` is given.
`lead_snp`	Center on this SNP ± flank.
`flank`	Flank size in bp.
`pip_min_size`	Minimum point size (for PIP 0).
`pip_max_size`	Maximum point size (for PIP 1).
`set_colors`	Colors for credible sets.
`nonsig_color`	Color for variants not in any credible set.
`show_pip_legend`	Show the PIP size legend.
`label_pip_above`	Label variants with PIP above this value.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas)
# Add simulated fine-mapping results
example_gwas$PIP <- runif(nrow(example_gwas))^4
example_gwas$PIP[which.min(example_gwas$P)] <- 0.95
example_gwas$credible_set <- NA
example_gwas$credible_set[example_gwas$PIP > 0.1] <- 1L
finemapping_plot(example_gwas, region = "chr1:1000000-50000000")
```

`forest_plot` *Forest plot of effect estimates*

Description

Draw a forest plot of effect estimates with confidence intervals, for example to compare a variant across cohorts, several lead variants from one study, or meta-analysis inputs. Estimates and standard errors are turned into confidence intervals; optionally the effects are exponentiated to display odds or hazard ratios.

Usage

```
forest_plot(
  data,
  effect = "BETA",
  se = "SE",
  label = NULL,
  group = NULL,
  ci = 0.95,
  exponentiate = FALSE,
  ref_line = if (exponentiate) 1 else 0,
  order_by = c("none", "effect", "label"),
  colors = NULL,
  point_size = 2.5,
  x_label = NULL,
  title = NULL
)
```

Arguments

`data`	A data.frame (or `gwas_data`) with effect and standard-error columns.
`effect`, `se`	Column names for the effect estimate and its standard error.
`label`	Column used for the row labels. Defaults to `SNP` when present, otherwise the row index.
`group`	Optional grouping column (e.g. cohort); groups are colored and offset within each label.
`ci`	Confidence level for the intervals (default 0.95).
`exponentiate`	If TRUE, plot `exp(effect)` on a log axis (odds/hazard ratios) and default the reference line to 1.
`ref_line`	Position of the vertical reference line.
`order_by`	Order rows by `"none"`, `"effect"`, or `"label"`.
`colors`	Optional colors for groups.
`point_size`	Point size.
`x_label`	Optional x-axis label.
`title`	Plot title.

Value

A ggplot object.

Examples

```
df <- data.frame(
  SNP = c("rs1", "rs2", "rs3", "rs4"),
  BETA = c(0.20, -0.10, 0.35, 0.05),
  SE = c(0.05, 0.04, 0.08, 0.03)
)
forest_plot(df)

# Odds ratios, ordered by effect
forest_plot(df, exponentiate = TRUE, order_by = "effect")
```

---

gene_track *Gene annotation track*

Description

Create a standalone gene annotation panel that can be composed with any ggwas plot using patchwork. Genes are drawn as directional bodies with strand arrows and labels; when `exons` are supplied, each gene is rendered as an intron backbone with exon boxes and a strand arrow.

Usage

```
gene_track(
  genes,
  region,
  exons = NULL,
  highlight_genes = NULL,
  highlight_color = "#E74C3C",
  label_size = 2.5,
  track_color = "#1A5276",
  show_strand = TRUE,
  max_genes = 50
)
```

Arguments

`genes`	Gene models as a GenomicRanges::GRanges object, or a `TxDb` (**GenomicFeatures**) or `EnsDb` (**ensembldb**) annotation object. Labels are taken from the first metadata column named `gene_name`, `symbol`, `gene` or `Name`, falling back to `gene_id` and then to `names(genes)`. A `TxDb` only carries gene IDs; add a `gene_name` column to `GenomicFeatures::genes(txdb)` to show symbols (see the vignette). GTF and GFF3 files can be read with `rtracklayer::import()`.

`region` The region to draw: a GenomicRanges::GRanges of length 1, or a string such as `"chr6:25000000-34000000"`.

`exons` Optional exon structure: a `GRangesList` of exons split by gene ID (as returned by `GenomicFeatures::exonsBy(txdb, by = "gene")`), a `GRanges` with a `gene_id` or gene name column, or `TRUE` to take exons from `genes` when it is a `TxDb` or `EnsDb`. Exons are matched to genes by gene ID.

`highlight_genes` Character vector of gene names to highlight.

`highlight_color` Color for highlighted genes.

`label_size` Text size for gene labels.

`track_color` Default color for gene bodies.

`show_strand` If TRUE, draw arrows indicating gene direction.

`max_genes` Maximum number of genes to display. The longest genes are kept when the limit is exceeded.

Value

A ggplot object. Compose with a Manhattan or locus plot using `patchwork::wrap_plots()`.

See Also

`locus_plot()`, which adds this track under a regional plot.

Examples

```
library(GenomicRanges)
genes <- GRanges(
  c("chr1:1000000-2000000:+", "chr1:5000000-6000000:-",
    "chr1:8000000-9000000:+"),
  gene_name = c("GeneA", "GeneB", "GeneC")
)

# Gene-body track
gene_track(genes, region = "chr1:1-10000000")

# With exon structure, one GRanges per gene
exons <- GRangesList(
  GeneA = GRanges(c("chr1:1000000-1100000", "chr1:1500000-1700000")),
  GeneB = GRanges(c("chr1:5000000-5200000", "chr1:5600000-5900000")),
  GeneC = GRanges("chr1:8100000-8400000")
)
names(genes) <- names(exons)
gene_track(genes, "chr1:1-10000000", exons = exons)
```

`genetic_correlation` *Genetic correlation matrix*

Description

Heatmap of genetic correlations (rg) between traits, typically from LD Score Regression (LDSC). Includes significance indicators and hierarchical clustering of traits.

Usage

```
genetic_correlation(
  rg_matrix,
  p_matrix = NULL,
  sig_threshold = 0.05,
  sig_marker = "*",
  palette = "RdBu",
  cluster = TRUE,
  rg_range = c(-1, 1),
  cell_size = 2.8,
  show_values = TRUE,
  show_diagonal = FALSE,
  title = NULL
)
```

Arguments

`rg_matrix`	A symmetric matrix of genetic correlations, or a data.frame with columns: trait1, trait2, rg, se, p.
`p_matrix`	Optional matrix of p-values (same dimensions as rg_matrix). If `rg_matrix` is a data.frame with a `p` column, this is extracted automatically.
`sig_threshold`	P-value threshold for significance markers.
`sig_marker`	Character to mark significant correlations.
`palette`	Color palette: "RdBu" (default), "PRGn", "PiYG", or a vector of colors.
`cluster`	If TRUE, reorder traits by hierarchical clustering.
`rg_range`	Range for the color scale (default c(-1, 1)).
`cell_size`	Size of the text inside cells.
`show_values`	Show rg values inside cells.
`show_diagonal`	Show the diagonal (always 1.0).
`title`	Plot title.

Value

A ggplot object.

Examples

```
# Simulated genetic correlation matrix
traits <- c("BMI", "Height", "WHR", "T2D", "CAD")
rg <- matrix(c(
  1.0,  -0.1,  0.6,  0.4,  0.2,
 -0.1,   1.0, -0.3, -0.1,  0.0,
  0.6,  -0.3,  1.0,  0.3,  0.15,
  0.4,  -0.1,  0.3,  1.0,  0.5,
  0.2,   0.0,  0.15, 0.5,  1.0
), nrow = 5, dimnames = list(traits, traits))
genetic_correlation(rg)

# With clustering
genetic_correlation(rg, cluster = TRUE)

# PRGn palette, no clustering
genetic_correlation(rg, palette = "PRGn", cluster = FALSE)

# PiYG palette
genetic_correlation(rg, palette = "PiYG")
```

---

get_loci *Get significant loci*

Description

Extract variants below a p-value threshold, with optional clumping to return only independent lead SNPs.

Usage

```
get_loci(data, p_threshold = 5e-08, clump = TRUE, clump_window = 1e+06)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`p_threshold`	P-value threshold (default 5e-8).
`clump`	If TRUE, apply window-based clumping.
`clump_window`	Clumping window in bp (default 1 Mb).

Value

A data.frame of significant variants.

Examples

```
data(example_gwas)
get_loci(example_gwas, p_threshold = 0.001)
```

gwas_palette *GWAS Color Palettes*

Description

Pre-defined color palettes for GWAS visualizations. The two-color palettes used by default for chromosomes stay distinguishable under the common forms of color blindness: "default" (dark blue and light teal, used by `manhattan_plot()`, `miami_plot()`, and `circular_manhattan()`) and "colorblind" (Okabe-Ito blue and orange, used by `scale_color_gwas()` and `gwas_summary()`). The journal palettes ("nature", "science", "lancet", "nejm", "aaas") and "brewer_set1" and "brewer_paired" match common figure styles, but they contain both red and green and are not colorblind-safe.

Usage

```
gwas_palette(name = "default", n = NULL, type = "alternating")
```

Arguments

`name`	Palette name. One of: "default", "colorblind", "vibrant", "pastel", "dark", "nature", "science", "lancet", "nejm", "aaas", "brewer_set1", "brewer_set2", "brewer_paired", "brewer_dark2".
`n`	Number of colors to return. If NULL, returns the full palette.
`type`	For chromosome palettes: "alternating" (2 colors) or "distinct" (one per chromosome).

Value

A character vector of hex colors.

Examples

```
data(example_gwas)
gwas_palette("colorblind")
gwas_palette("nature", n = 5)
manhattan_plot(example_gwas, colors = gwas_palette("vibrant"))
```

gwas_palettes *List available GWAS palettes*

Description

List available GWAS palettes

Usage

```
gwas_palettes()
```

Value

A character vector of palette names.

Examples

```
gwas_palettes()
```

---

gwas_preset *GWAS plot presets*

Description

Apply a preset configuration to any ggwas function. Presets combine a theme, color palette, and sizing appropriate for the context.

Usage

```
gwas_preset(preset = "publication")
```

Arguments

`preset`	One of: "publication", "presentation", "poster", "exploratory".

Value

A list with components: `theme`, `colors`, `point_size`, `alpha`, `label_size`.

Examples

```
data(example_gwas)
p <- gwas_preset("publication")
manhattan_plot(example_gwas, colors = p$colors, point_size = p$point_size) +
  p$theme
```

---

gwas_summary *GWAS Summary Panel*

Description

A multi-panel dashboard combining Manhattan plot, QQ plot, top hits table, and summary statistics into a single publication-ready figure.

Usage

```
gwas_summary(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  panels = c("manhattan", "qq", "top_hits", "density"),
  n_top = 5,
  genome_wide = 5e-08,
  suggestive = 1e-05,
  colors = NULL,
  palette = "colorblind",
  label_top_n = 3,
  title = NULL,
  theme_fn = NULL,
  layout = "auto"
)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`panels`	Character vector of panels to include. Options: "manhattan", "qq", "top_hits", "density", "stats".
`n_top`	Number of top hits to display in the table.
`genome_wide`	Genome-wide significance threshold.
`suggestive`	Suggestive significance threshold.
`colors`	Two-color vector for Manhattan chromosome colors.
`palette`	Palette name from `gwas_palette()`.
`label_top_n`	Label top N SNPs on Manhattan.
`title`	Overall title.
`theme_fn`	Theme function to apply (default `theme_gwas()`).
`layout`	Layout arrangement: "auto", "wide", or "tall".

Value

A patchwork composition.

Examples

```
data(example_gwas)

# Full summary dashboard
gwas_summary(example_gwas)
```

```
# Manhattan and QQ only
gwas_summary(example_gwas, panels = c("manhattan", "qq"))
```

`highlight_regions` *Add highlighted regions to a Manhattan plot*

Description

Draws colored rectangular bands behind specified genomic regions. Useful for marking known GWAS loci, MHC region, or custom regions of interest.

Usage

```
highlight_regions(
  plt,
  regions,
  color = "#FFD700",
  alpha = 0.15,
  label_size = 2.5,
  label_y = "top",
  border = FALSE,
  border_color = "grey40"
)
```

Arguments

`plt`	A ggplot object from `manhattan_plot()` or similar.
`regions`	Regions as a GenomicRanges::GRanges object. Optional metadata columns: `label`, `color`, `alpha`.
`color`	Default fill color for regions (if not specified per region).
`alpha`	Default transparency.
`label_size`	Font size for region labels.
`label_y`	Y-axis position for labels ("top" or numeric value).
`border`	If TRUE, draw a border around regions.
`border_color`	Border color.

Value

The ggplot object with region highlights added.

Examples

```
data(example_gwas)
plt <- manhattan_plot(example_gwas)
regions <- GenomicRanges::GRanges(
  c("chr6:25000000-35000000", "chr1:5000000-20000000"),
  label = c("MHC", "1p36")
)
highlight_regions(plt, regions)
```

---

journal_themes *Journal-specific themes for GWAS plots*

Description

Publication-ready themes matching the figure style guides of major journals. Each theme sets appropriate font family, sizes, line weights, and spacing.

Usage

```
theme_nature(base_size = 7, base_family = NULL)

theme_science(base_size = 8, base_family = NULL)

theme_plos(base_size = 10, base_family = NULL)

theme_cell(base_size = 7, base_family = NULL)

theme_presentation(base_size = 16, base_family = "")

theme_poster(base_size = 20, base_family = "")
```

Arguments

`base_size`	Base font size in pt.
`base_family`	Base font family.

Details

`theme_nature()`: Helvetica/Arial 5-7 pt, minimal gridlines, thin axes (0.25 pt), compact margins. Matches Nature's requirement for figures at 89 mm (single column) or 183 mm (double column) width.

`theme_science()`: Helvetica 6-8 pt, classic axes (no panel border), thicker axis lines (0.5 pt), no grid. Science figures are narrow (55 mm single, 120 mm double, 174 mm full width).

`theme_plos()`: Arial 8-12 pt (larger than Nature/Science), panel border, centered title. PLOS requires figures at 13.2 cm single column width, 300 dpi minimum.

`theme_cell()`: Arial 6-8 pt, minimalist with very thin axes (0.2 pt), no grid, tight margins. Cell Press figures use 85 mm single and 178 mm full width.

`theme_presentation()`: Large fonts (16 pt base), thick axes, high contrast for slides projected at a distance.

`theme_poster()`: Extra-large fonts (20 pt base), thick axes, readable from 1-2 meters at a conference poster.

Value

A ggplot2 theme object.

Examples

```
data(example_gwas)
manhattan_plot(example_gwas) + theme_nature()
```

---

locus_plot *Locus zoom plot*

Description

Create a regional association plot centered on a genomic region or lead SNP, with optional LD coloring and gene annotation track.

Usage

```
locus_plot(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  region = NULL,
  lead_snp = NULL,
  flank = 5e+05,
  ld = NULL,
  ld_colors = c("#2166AC", "#67A9CF", "#78C679", "#F4A582", "#D73027"),
  ld_breaks = c(0, 0.2, 0.4, 0.6, 0.8, 1),
  genes = NULL,
  exons = NULL,
  gene_height = 0.25,
  point_size = 2,
  lead_snp_shape = 23,
  lead_snp_size = 4,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`region`	The region to plot: a GenomicRanges::GRanges of length 1, or a string such as `"chr1:1000000-20000000"`. Not needed when `lead_snp` is given.
`lead_snp`	SNP ID to center the region around.
`flank`	Flank size in bp around the lead SNP (default 500kb).
`ld`	Named numeric vector of LD (r2) values with the lead SNP. Names are SNP IDs, values are r2.
`ld_colors`	Colors for LD bins (5 levels).
`ld_breaks`	Breaks for LD color bins.
`genes`	Optional gene models drawn as a track under the plot: a GenomicRanges::GRanges, `TxDb` or `EnsDb` object (see `gene_track()`).
`exons`	Optional exon structure for the gene track (see `gene_track()`).
`gene_height`	Proportion of plot height for the gene track.
`point_size`	Point size.
`lead_snp_shape`	Shape for the lead SNP marker.
`lead_snp_size`	Size for the lead SNP marker.
`title`	Plot title.

Value

A ggplot object, or a patchwork composition when `genes` is provided.

Examples

```
data(example_gwas, package = "ggwas")
locus_plot(example_gwas, region = "chr1:1000000-20000000")

# With a gene track
genes <- GenomicRanges::GRanges(
  c("chr1:2000000-4000000:+", "chr1:9000000-12000000:-"),
  gene_name = c("GeneA", "GeneB")
)
locus_plot(example_gwas, region = "chr1:1000000-20000000", genes = genes)
```

maf_filter *Filter variants by minor allele frequency*

Description

Keep only variants with allele frequency within a specified range. Requires the AF column.

Usage

```
maf_filter(data, min_maf = 0.01, max_maf = 0.5)
```

Arguments

data	A gwas_data object or data.frame with an AF column.
min_maf	Minimum minor allele frequency (default 0.01).
max_maf	Maximum minor allele frequency (default 0.5).

Value

A filtered gwas_data object.

Examples

```
data(example_gwas)
common <- maf_filter(example_gwas, min_maf = 0.05)
```

manhattan_genes *Add gene labels to a Manhattan plot*

Description

A convenience wrapper that annotates a dataset and then creates a Manhattan plot with gene names as labels instead of SNP IDs.

Usage

```
manhattan_genes(
  data,
  genes,
  gene_p_threshold = 1e-05,
  gene_top_n = 10,
  gene_max_distance = 5e+05,
  arrow = FALSE,
  arrow_color = "grey30",
  label_size = 3,
  label_face = "italic",
  ...
)
```

Arguments

data	A gwas_data object or data.frame with GWAS results.
genes	Gene models as a GRanges, TxDb or EnsDb object (see annotate_genes()).
gene_p_threshold	P-value threshold for gene annotation.
gene_top_n	Number of top genes to label.
gene_max_distance	Maximum distance to nearest gene.
arrow	If TRUE, use annotation arrows instead of text labels.
arrow_color	Color of annotation arrows.
label_size	Font size for gene labels.
label_face	Font face for gene labels ("italic", "bold", "bold.italic").
...	Additional arguments passed to manhattan_plot().

Value

A ggplot object.

Examples

```
data(example_gwas)
genes <- GenomicRanges::GRanges(
  c("chr1:1000000-2000000", "chr2:50000000-52000000",
    "chr5:80000000-85000000", "chr7:30000000-35000000",
    "chr11:60000000-65000000"),
  gene_name = c("GENE_A", "GENE_B", "GENE_C", "GENE_D", "GENE_E")
)
manhattan_genes(example_gwas, genes = genes, gene_top_n = 5)
```

---

manhattan_plot *Manhattan plot*

Description

Create a genome-wide Manhattan plot from GWAS summary statistics. Returns a ggplot2 object for further customization.

Usage

```
manhattan_plot(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
```

```
  colors = c("#1A5276", "#76D7C4"),
  point_size = 0.8,
  alpha = 1,
  genome_wide = 5e-08,
  suggestive = 1e-05,
  threshold_colors = c("#E74C3C", "#3498DB"),
  highlight_snps = NULL,
  highlight_color = "#E74C3C",
  highlight_size = 2,
  label_snps = NULL,
  label_top_n = NULL,
  label_column = "SNP",
  downsample = TRUE,
  downsample_n = 2e+05,
  chromosomes = NULL,
  chr_labels = NULL,
  y_metric = "p",
  y_limit = NULL,
  y_truncate = NULL,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame with GWAS results.
`chr`, `bp`, `p`, `snp`	Column name overrides (auto-detected if NULL).
`colors`	Two-color vector for alternating chromosomes.
`point_size`	Point size.
`alpha`	Point transparency.
`genome_wide`	Genome-wide significance threshold (p-value).
`suggestive`	Suggestive significance threshold.
`threshold_colors`	Colors for significance lines.
`highlight_snps`	Character vector of SNP IDs to highlight.
`highlight_color`	Color for highlighted SNPs.
`highlight_size`	Size for highlighted points.
`label_snps`	Character vector of SNP IDs to label.
`label_top_n`	Label the top N most significant SNPs.
`label_column`	Column to use for label text.
`downsample`	Enable smart downsampling for large datasets.
`downsample_n`	Target number of points after downsampling.
`chromosomes`	Subset of chromosomes to plot (integer vector).

`chr_labels`	Custom chromosome labels. Options: NULL (all labels), `"odd"` (only odd-numbered chromosomes labeled), or a character vector of labels (same length as displayed chromosomes).
`y_metric`	What to plot on the y-axis. `"p"` (default) shows -log10(p). `"beta_min"` shows the lower confidence bound of the absolute effect size: |beta| - 2*SE. Variants whose confidence interval overlaps zero are excluded. This highlights variants with large, robust effect sizes rather than just small p-values.
`y_limit`	Upper y-axis limit for -log10(p).
`y_truncate`	Break the y-axis for datasets with extreme peaks. A single value shows 0 to that value at full scale and compresses everything above it into a band above the break (nothing is dropped), e.g. `y_truncate = 15`. A vector of two values `c(break_from, resume_at)` instead cuts out the middle region: `y_truncate = c(15, 50)` shows 0-15 at full scale, drops 15-50, then shows 50+ above the break. Units are -log10(p).
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")

# Basic Manhattan plot
manhattan_plot(example_gwas)

# Label top hits with a different palette
manhattan_plot(example_gwas, label_top_n = 5, colors = gwas_palette("vibrant"))

# Nature journal style
manhattan_plot(example_gwas, label_top_n = 3) + theme_nature()

# Lancet palette with Science theme
manhattan_plot(example_gwas, colors = gwas_palette("lancet")) + theme_science()

# Subset chromosomes
manhattan_plot(example_gwas, chromosomes = seq_len(10))

# NEJM palette
manhattan_plot(example_gwas, colors = gwas_palette("nejm"), label_top_n = 3)

# Broken y-axis: cut 10-50, show 0-10 and 50+
manhattan_plot(example_gwas, y_truncate = c(10, 50))

# Single value: auto-detect resume point
manhattan_plot(example_gwas, y_truncate = 10)

# Custom significance threshold (Bonferroni for 500k SNPs)
```

```
manhattan_plot(example_gwas, genome_wide = 0.05 / 500000)

# No threshold lines
manhattan_plot(example_gwas, genome_wide = NULL, suggestive = NULL)

# Label only odd chromosomes (less crowded x-axis)
manhattan_plot(example_gwas, chr_labels = "odd")

# Effect-size confidence bound (|beta| - 2*SE)
manhattan_plot(example_gwas, y_metric = "beta_min")
```

`merge_gwas` *Merge multiple GWAS datasets*

Description

Combine results from multiple GWAS into a single data.frame with a `study` column identifying the source. Useful for multi-trait comparisons and meta-analysis visualization.

Usage

```
merge_gwas(..., by = NULL)
```

Arguments

`...`	Named `gwas_data` objects or data.frames. Names become the `study` column values.
`by`	Columns to match variants across studies. Default uses SNP if available, otherwise CHR + BP.

Value

A data.frame with an added `study` column.

Examples

```
data(example_gwas)
trait1 <- example_gwas
trait2 <- example_gwas
trait2$P <- runif(nrow(trait2))
merged <- merge_gwas(BMI = trait1, Height = trait2)
head(merged)
```

`miami_plot` *Miami plot (mirrored Manhattan)*

Description

Create a Miami plot showing two GWAS results as mirrored Manhattan plots. The top panel shows one study with -log10(p) going up, and the bottom panel shows another study with -log10(p) going down.

Usage

```
miami_plot(
  top,
  bottom,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  colors = c("#1A5276", "#76D7C4"),
  genome_wide = 5e-08,
  suggestive = 1e-05,
  top_highlight = NULL,
  bottom_highlight = NULL,
  top_label = NULL,
  bottom_label = NULL,
  top_title = NULL,
  bottom_title = NULL,
  downsample = TRUE,
  downsample_n = 2e+05,
  title = NULL
)
```

Arguments

`top`	A `gwas_data` object or data.frame for the top panel.
`bottom`	A `gwas_data` object or data.frame for the bottom panel.
`chr`, `bp`, `p`, `snp`	Column name overrides.
`colors`	Two-color vector for alternating chromosomes.
`genome_wide`	Genome-wide significance threshold.
`suggestive`	Suggestive significance threshold.
`top_highlight`, `bottom_highlight`	SNP IDs to highlight in each panel.
`top_label`, `bottom_label`	SNP IDs to label in each panel.

`top_title`, `bottom_title`
: Y-axis titles for each panel.

`downsample`
: Enable smart downsampling.

`downsample_n`
: Target points per panel.

`title`
: Overall plot title.

Value

A ggplot object composed via patchwork.

Examples

```
data(example_gwas, package = "ggwas")

# Discovery vs replication
miami_plot(example_gwas, example_gwas,
  top_title = "Discovery", bottom_title = "Replication")

# Different colors
miami_plot(example_gwas, example_gwas,
  colors = gwas_palette("nature"),
  top_title = "Study 1", bottom_title = "Study 2")
```

`multitrait_manhattan` *Multi-trait Manhattan plot*

Description

Overlay results from multiple GWAS traits or studies on a single Manhattan plot. Each trait is shown in a different color and optionally a different shape, enabling visual identification of shared and trait-specific loci (pleiotropy).

Usage

```
multitrait_manhattan(
  ...,
  chr = NULL,
  bp = NULL,
  p = NULL,
  snp = NULL,
  colors = "nature",
  shapes = NULL,
  point_size = 0.8,
  alpha = 0.7,
  genome_wide = 5e-08,
  suggestive = 1e-05,
  show_legend = TRUE,
  legend_position = "top",
```

```
  highlight_shared = TRUE,
  shared_color = "#9B59B6",
  shared_size = 2.5,
  label_shared_n = NULL,
  downsample = TRUE,
  downsample_n = 150000,
  title = NULL
)
```

Arguments

`...`	Named `gwas_data` objects or data.frames. Names are used as trait labels. Alternatively, pass a single named list.
`chr`, `bp`, `p`, `snp`	Column name overrides applied to all datasets.
`colors`	Named vector of colors per trait, or a palette name from `gwas_palette()`.
`shapes`	Named vector of point shapes per trait (integers). If NULL, all use shape 16 (circle).
`point_size`	Point size (can be a named vector per trait).
`alpha`	Point transparency.
`genome_wide`	Genome-wide significance threshold.
`suggestive`	Suggestive significance threshold.
`show_legend`	Show trait legend.
`legend_position`	Legend position.
`highlight_shared`	Highlight SNPs significant in multiple traits.
`shared_color`	Color for shared significant SNPs.
`shared_size`	Size for shared significant SNPs.
`label_shared_n`	Label top N shared significant SNPs.
`downsample`	Enable smart downsampling per trait.
`downsample_n`	Target points per trait.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas)
# Simulate two traits
trait1 <- example_gwas
trait2 <- example_gwas
trait2$P <- runif(nrow(trait2))^3
multitrait_manhattan(Trait1 = trait1, Trait2 = trait2)
```

`phewas_plot` *PheWAS plot*

Description

Visualize one variant tested across many phenotypes, grouped by phenotype category. The standard figure for biobank-scale phenome-wide association studies (UK Biobank, FinnGen, BioBank Japan).

Usage

```
phewas_plot(
  data,
  p = "p",
  phenotype = "phenotype",
  category = "category",
  beta = "beta",
  p_threshold = 5e-08,
  colors = NULL,
  point_size = 2,
  alpha = 0.8,
  label_top_n = 5,
  label_column = NULL,
  show_categories = TRUE,
  category_label_angle = 45,
  direction_shape = TRUE,
  title = NULL
)
```

Arguments

`data`	A data.frame with columns: `phenotype`, `p` (p-value), and `category` (phenotype group). Optional: `beta`, `description`.
`p`	Column name for p-values.
`phenotype`	Column name for phenotype labels.
`category`	Column name for phenotype categories.
`beta`	Column name for effect sizes (used for direction triangles).
`p_threshold`	Significance threshold line.
`colors`	Named vector of colors per category, or a palette name.
`point_size`	Point size.
`alpha`	Point transparency.
`label_top_n`	Label the top N most significant phenotypes.
`label_column`	Column for label text (default: phenotype).

`show_categories`
: Show category labels on x-axis.

`category_label_angle`
: Rotation angle for category labels.

`direction_shape`
: If TRUE and `beta` is provided, use triangles pointing up (positive) or down (negative) instead of circles.

`title`
: Plot title.

Value

A ggplot object.

Examples

```
phewas_data <- data.frame(
  phenotype = paste0("Pheno_", seq_len(50)),
  p = 10^(-runif(50, 0, 8)),
  category = rep(c("Metabolic", "Immune", "Neuro", "Cardio", "Other"), 10),
  beta = rnorm(50, 0, 0.2)
)
phewas_plot(phewas_data)
```

`pvalue_heatmap` *Genome-wide p-value heatmap*

Description

A compact, binned heatmap of association signals across the genome. Each cell represents a genomic bin colored by the summary statistic (minimum p-value, median, or count of significant variants). Handles 10M+ SNPs efficiently through pre-aggregation.

Usage

```
pvalue_heatmap(
  data,
  chr = NULL,
  bp = NULL,
  p = NULL,
  bin_size = 1e+06,
  summary_fun = "min",
  palette = "viridis",
  na_color = "grey95",
  threshold = 5e-08,
  chromosomes = NULL,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`chr`, `bp`, `p`	Column name overrides.
`bin_size`	Bin size in base pairs (default 1 Mb).
`summary_fun`	How to summarize p-values per bin: "min" (default), "median", or "count_sig" (count of variants below `threshold`).
`palette`	Color palette: "viridis", "magma", "inferno", or "plasma".
`na_color`	Color for empty bins.
`threshold`	Significance threshold for count_sig mode and for marking significant bins.
`chromosomes`	Subset of chromosomes to show.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")

# Default (viridis)
pvalue_heatmap(example_gwas, bin_size = 10000)

# Magma palette
pvalue_heatmap(example_gwas, bin_size = 10000, palette = "magma")

# Inferno palette
pvalue_heatmap(example_gwas, bin_size = 10000, palette = "inferno")

# Count significant variants per bin
pvalue_heatmap(example_gwas, bin_size = 10000, summary_fun = "count_sig")
```

`qq_plot` *QQ plot for GWAS p-values*

Description

Create a quantile-quantile plot with confidence bands and genomic inflation factor.

Usage

```
qq_plot(
  data,
  p = NULL,
  ci = 0.95,
  ci_fill = "grey80",
  ci_alpha = 0.3,
  show_lambda = TRUE,
  lambda_position = "topleft",
  group = NULL,
  point_size = 0.8,
  point_color = "#1A5276",
  alpha = 1,
  line_color = "#E74C3C",
  downsample = TRUE,
  downsample_n = 1e+05,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame, or a numeric vector of p-values.
`p`	Column name for p-values (auto-detected if NULL).
`ci`	Confidence level for bands (NULL to disable).
`ci_fill`	Fill color for confidence band.
`ci_alpha`	Transparency of confidence band.
`show_lambda`	Show genomic inflation factor annotation.
`lambda_position`	Position for lambda annotation: "topleft" or "bottomright".
`group`	Column name to stratify QQ plot by groups.
`point_size`	Point size.
`point_color`	Point color (ignored if group is specified).
`alpha`	Point transparency.
`line_color`	Color of the identity line.
`downsample`	Enable downsampling of dense lower-left region.
`downsample_n`	Target number of points.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")

# Basic QQ with lambda and confidence band
qq_plot(example_gwas, show_lambda = TRUE, ci = 0.95)

# Without confidence band
qq_plot(example_gwas, show_lambda = TRUE, ci = NULL)

# Stratify by allele frequency
example_gwas$maf_bin <- cut(example_gwas$AF, c(0, 0.05, 0.2, 0.5),
                            labels = c("Rare", "Low", "Common"))
qq_plot(example_gwas, group = "maf_bin")

# Nature theme
qq_plot(example_gwas, show_lambda = TRUE) + theme_nature()
```

---

read_gcta_mlma *Read GCTA MLMA results*

Description

Read GCTA MLMA results

Usage

```
read_gcta_mlma(file, ...)
```

Arguments

`file`	Path to a GCTA .mlma file.
`...`	Additional arguments passed to `data.table::fread()`.

Value

A `gwas_data` object.

Examples

```
f <- system.file("extdata", "example_gcta.mlma", package = "ggwas")
gwas <- read_gcta_mlma(f)
gwas
```

read_gemma *Read GEMMA association results*

Description

Read GEMMA association results

Usage

```
read_gemma(file, p_column = "p_wald", ...)
```

Arguments

file	Path to a GEMMA .assoc.txt file.
p_column	Which p-value column to use: "p_wald", "p_lrt", or "p_score".
...	Additional arguments passed to data.table::fread().

Value

A gwas_data object.

Examples

```
f <- system.file("extdata", "example_gemma.assoc.txt", package = "ggwas")
gwas <- read_gemma(f)
gwas
```

read_gwas_table *Read GWAS summary statistics from any tabular file*

Description

Reads a GWAS results file with automatic column detection. Supports any delimited text file. Column names are matched against common GWAS naming conventions.

Usage

```
read_gwas_table(
  file,
  chr = NULL,
  bp = NULL,
  snp = NULL,
  p = NULL,
  beta = NULL,
  se = NULL,
  a1 = NULL,
```

```
  a2 = NULL,
  af = NULL,
  n = NULL,
  info = NULL,
  sep = "auto",
  log_p = FALSE,
  ...
)
```

Arguments

`file`	Path to the input file.
`chr`, `bp`, `snp`, `p`, `beta`, `se`, `a1`, `a2`, `af`, `n`, `info`	Optional column name overrides.
`sep`	Column separator. "auto" uses `data.table::fread()` auto-detection.
`log_p`	If TRUE, the p-value column contains -log10(p) values.
`...`	Additional arguments passed to `data.table::fread()`.

Value

A `gwas_data` object.

Examples

```
f <- system.file("extdata", "example_plink.assoc", package = "ggwas")
gwas <- read_gwas_table(f)
gwas
```

`read_plink_assoc` *Read PLINK association results*

Description

Read PLINK association results

Usage

```
read_plink_assoc(file, ...)
```

Arguments

`file`	Path to a PLINK .assoc file.
`...`	Additional arguments passed to `data.table::fread()`.

Value

A `gwas_data` object.

Examples

```
f <- system.file("extdata", "example_plink.assoc", package = "ggwas")
gwas <- read_plink_assoc(f)
gwas
```

---

read_plink_linear *Read PLINK linear regression results*

Description

Read PLINK linear regression results

Usage

```
read_plink_linear(file, test = "ADD", ...)
```

Arguments

file	Path to a PLINK .assoc.linear file.
test	Which test to keep (default "ADD" for additive model).
...	Additional arguments passed to data.table::fread().

Value

A gwas_data object.

Examples

```
f <- system.file("extdata", "example_plink.assoc.linear", package = "ggwas")
gwas <- read_plink_linear(f)
gwas
```

---

read_plink_logistic *Read PLINK logistic regression results*

Description

Read PLINK logistic regression results

Usage

```
read_plink_logistic(file, test = "ADD", ...)
```

Arguments

file	Path to a PLINK .assoc.logistic file.
test	Which test to keep (default "ADD" for additive model).
...	Additional arguments passed to data.table::fread().

Value

A gwas_data object.

Examples

```
f <- system.file("extdata", "example_plink.assoc.logistic", package = "ggwas")
gwas <- read_plink_logistic(f)
gwas
```

---

read_regenie *Read REGENIE results*

Description

Read REGENIE results

Usage

```
read_regenie(file, ...)
```

Arguments

file	Path to a .regenie results file.
...	Additional arguments passed to data.table::fread().

Value

A gwas_data object.

Examples

```
f <- system.file("extdata", "example_regenie.regenie", package = "ggwas")
gwas <- read_regenie(f)
gwas
```

`scale_color_chromosome` *Chromosome color scale*

Description

Alternating color scale for chromosomes in Manhattan plots.

Usage

```
scale_color_chromosome(
  colors = c("#1A5276", "#76D7C4"),
  chromosomes = NULL,
  ...
)

scale_fill_chromosome(
  colors = c("#1A5276", "#76D7C4"),
  chromosomes = NULL,
  ...
)
```

Arguments

`colors`	Two-element character vector of alternating colors.
`chromosomes`	Chromosome codes present in the data, in the order they appear along the genome. Colors alternate across this set, so any organism is handled (e.g. cattle with 29 autosomes plus X coded as 30). When `NULL`, defaults to the human coding (autosomes 1-22 plus X/Y/XY/MT as 23-26).
`...`	Additional arguments passed to `ggplot2::scale_color_manual()`.

Value

A ggplot2 color scale.

Examples

```
data(example_gwas)
manhattan_plot(example_gwas, colors = c("#E64B35", "#4DBBD5"))
```

scale_color_gwas *Chromosome color scale*

Description

Alternating colors for chromosomes. The default palette ("colorblind") is colorblind-safe; see `gwas_palette()` for the other options.

Usage

```
scale_color_gwas(palette = "colorblind", chromosomes = NULL, ...)

scale_fill_gwas(palette = "colorblind", chromosomes = NULL, ...)
```

Arguments

`palette`	Palette name from `gwas_palette()`.
`chromosomes`	Chromosome codes present in the data, in genomic order. Colors cycle across this set, so non-human karyotypes are handled. When `NULL`, defaults to the human coding (26 chromosomes).
`...`	Additional arguments passed to `ggplot2::scale_color_manual()`.

Value

A ggplot2 color scale.

Examples

```
scale_color_gwas("nature")
scale_fill_gwas("colorblind")
```

set_sex_chr_map *Configure sex chromosome mapping*

Description

ggwas stores chromosomes as integer codes. By default it uses human conventions, where chromosome 23 is labeled "X", 24 is "Y", 25 is "XY" and 26 is "MT". For other organisms, pass the genome's `Seqinfo` object (**GenomeInfoDb**) and the codes are taken from its sequence names: numbered chromosomes keep their number, and named ones (X, Y, Z, W, MT, ...) are numbered after the last autosome, in `Seqinfo` order. Unplaced scaffolds are ignored.

Usage

```
set_sex_chr_map(map = .default_sex_chr_map)
```

Arguments

`map`	A `Seqinfo` object, or any object that has one (`TxDb`, `EnsDb`, `BSgenome`); or a named integer vector mapping labels to integers, e.g. `c(X = 20L, Y = 21L)`. Pass `NULL` to disable the mapping (chromosomes displayed as plain numbers).

Value

The previous mapping (invisibly).

Examples

```
library(GenomeInfoDb)

# Mouse (GRCm39): 19 autosomes, then X, Y and MT. With internet access,
# take the chromosomes from NCBI instead of typing them:
#   mouse <- keepStandardChromosomes(Seqinfo(genome = "GRCm39"))
mouse <- Seqinfo(c(1:19, "X", "Y", "MT"), genome = "GRCm39")
old <- set_sex_chr_map(mouse)
getOption("ggwas.sex_chr_map")

# Chicken (GRCg6a): W and Z follow the last numbered chromosome, 33.
# With internet access (unplaced scaffolds are ignored):
#   chicken <- Seqinfo(genome = "GRCg6a")
chicken <- Seqinfo(c(1:28, 30:33, "W", "Z", "MT"), genome = "GRCg6a")
set_sex_chr_map(chicken)

# Disable: all chromosomes shown as numbers
set_sex_chr_map(NULL)

# Restore the previous (here the human default) mapping
set_sex_chr_map(old)
```

`snp_density` *SNP density karyogram*

Description

Visualize the distribution of genotyped or imputed variants across chromosomes. Two rendering styles are available: `"heatmap"` (default) bins variants and colors tiles by count, while `"points"` draws individual variant positions on chromosome outlines, where density is visible through natural clustering. Optionally marks centromere positions when `centromeres` are provided.

Usage

```
snp_density(
  data,
  chr = NULL,
```

```
  bp = NULL,
  p = NULL,
  style = c("heatmap", "points"),
  bin_size = 1e+06,
  color_by = c("density", "significance", "uniform"),
  seqinfo = NULL,
  centromeres = NULL,
  palette = "viridis",
  point_size = 0.3,
  point_alpha = 0.5,
  show_centromeres = TRUE,
  downsample_n = 2e+05,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame with GWAS results.
`chr`, `bp`, `p`	Column name overrides.
`style`	Rendering style: `"heatmap"` for binned density tiles, or `"points"` for individual variant positions on chromosome outlines.
`bin_size`	Bin size in base pairs (default 1 Mb). Only used when `style = "heatmap"`.
`color_by`	For `style = "points"`: how to color dots. `"density"` (default) colors by local density, `"significance"` colors by -log10(p), or `"uniform"` for a single color.
`seqinfo`	Optional chromosome lengths as a `Seqinfo` object (**GenomeInfoDb**) or any object that has one, such as a `TxDb`, `EnsDb` or `BSgenome`. If NULL, chromosome lengths are inferred from the data.
`centromeres`	Optional centromere positions as a GenomicRanges::GRanges, one or more ranges per chromosome (for example the UCSC "Centromeres" track imported with **rtracklayer**).
`palette`	Color palette: "viridis", "magma", "inferno", or "plasma".
`point_size`	Point size for `style = "points"`.
`point_alpha`	Point transparency for `style = "points"`.
`show_centromeres`	If TRUE and `centromeres` are given, draw centromere markers.
`downsample_n`	Maximum number of points to plot in `"points"` style. Set to NULL to plot all variants (can be slow for >500k).
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")

# Heatmap style (default); chromosome lengths taken from the data
snp_density(example_gwas, bin_size = 5e6)

# Different palette
snp_density(example_gwas, bin_size = 5e6, palette = "magma")

# Points style with centromere markers (hg38 positions for chr1 and chr2)
cen <- GenomicRanges::GRanges(
  c("chr1:122026460-124932724", "chr2:92188146-94090557")
)
snp_density(example_gwas, style = "points", centromeres = cen)

# Points colored by significance
snp_density(example_gwas, style = "points", color_by = "significance")

# Full chromosome lengths from a Seqinfo, here taken from a TxDb
if (requireNamespace("TxDb.Hsapiens.UCSC.hg38.knownGene", quietly = TRUE)) {
  txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene::TxDb.Hsapiens.UCSC.hg38.knownGene
  snp_density(example_gwas, bin_size = 5e6,
    seqinfo = GenomeInfoDb::seqinfo(txdb))
}
```

---

theme_gwas *Minimal GWAS theme for ggplot2*

Description

A clean, publication-ready theme optimized for GWAS plots.

Usage

```
theme_gwas(base_size = 11, base_family = "")
```

Arguments

base_size	Base font size (default 11).
base_family	Base font family.

Value

A ggplot2 theme object.

Examples

```
data(example_gwas)
manhattan_plot(example_gwas) + theme_gwas()
```

`top_hits` *Extract top hits from GWAS results*

Description

Identifies independent significant loci using window-based clumping, optionally annotates with nearest gene, and returns a publication-ready table.

Usage

```
top_hits(
  data,
  p_threshold = 5e-08,
  clump_window = 1e+06,
  genes = NULL,
  max_gene_distance = 5e+05,
  n = 20
)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`p_threshold`	Significance threshold for lead SNPs.
`clump_window`	Window size in bp for clumping. Only the most significant SNP within each window is retained.
`genes`	Optional gene models as a GenomicRanges::GRanges, `TxDb` or `EnsDb` object. If provided, the nearest gene is added (see `annotate_genes()`).
`max_gene_distance`	Maximum distance (bp) to the nearest gene.
`n`	Maximum number of hits to return.

Value

A data.frame with columns: SNP, CHR, BP, P, BETA, SE, nearest_gene (if genes provided), gene_distance, cytoband.

Examples

```
data(example_gwas)
top_hits(example_gwas, p_threshold = 0.001, n = 10)
```

`trumpet_plot` *Trumpet plot: effect size versus allele frequency with power contours*

Description

Plot per-variant effect sizes against minor allele frequency and overlay statistical-power ("detection") contours. The contours trace the smallest effect a study of the given sample size can detect at a chosen significance level and power, producing the characteristic trumpet shape that flares toward rare variants. Points falling below all contours lie in the region a study is underpowered to discover.

Usage

```
trumpet_plot(
  data,
  beta = NULL,
  af = NULL,
  p = NULL,
  n = NULL,
  n_col = NULL,
  sig_level = 5e-08,
  power = c(0.5, 0.8),
  signed = FALSE,
  colors = c(significant = "#E64B35", nonsignificant = "#BDC3C7"),
  point_size = 1,
  alpha = 0.5,
  label_top_n = NULL,
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame with BETA and AF columns.
`beta`, `af`, `p`	Column name overrides.
`n`	Study sample size (single number). If NULL, taken from `n_col` or an `N` column (median).
`n_col`	Name of a per-variant sample-size column.
`sig_level`	Significance threshold for the power contours.
`power`	Numeric vector of power levels to draw contours for.
`signed`	If TRUE, plot signed effects (contours mirrored above and below zero); otherwise plot the absolute effect.
`colors`	Named vector with "significant" and "nonsignificant" colors.
`point_size`	Point size.
`alpha`	Point transparency.

`label_top_n`	Label the top N variants by significance.
`title`	Plot title.

Details

The power model assumes an additive test on a standardized quantitative trait (per-allele effect on a unit-variance scale). For each minor allele frequency f, the minimum detectable effect is $\beta_{min}(f) = \sqrt{\lambda/(2Nf(1-f))}$, where the non-centrality parameter $\lambda = (z_\alpha + z_{power})^2$.

Value

A ggplot object.

Examples

```
data(example_gwas)

# Effect size versus MAF with 50% and 80% power contours
trumpet_plot(example_gwas, n = 50000)

# Signed effects and a labelled top hit
trumpet_plot(example_gwas, n = 50000, signed = TRUE,
             p = "P", label_top_n = 3)
```

`validate_gwas_data` *Validate a gwas_data object*

Description

Validate a gwas_data object

Usage

```
validate_gwas_data(x)
```

Arguments

`x`	A gwas_data object.

Value

Invisible x (raises warnings/errors on invalid data).

Examples

```
data(example_gwas)
validate_gwas_data(example_gwas)
```

`value_manhattan` *Manhattan plot of an arbitrary per-feature value*

Description

Plot any numeric value assigned to a genomic feature (gene, transcript, window) along the genome in Manhattan style: features on the genomic x-axis, the value on the y-axis, alternating chromosome colors. Unlike `manhattan_plot()`, which expects GWAS p-values, this plots the value as-is, so it suits rankings, effect sizes, ranking differences between two transcriptome references, conservation scores, read counts, and similar per-feature metrics. Returns a ggplot2 object for further customization.

Usage

```
value_manhattan(
  data,
  value = NULL,
  chr = NULL,
  bp = NULL,
  id = NULL,
  annotation = NULL,
  id_type = c("gene_id", "symbol"),
  label = NULL,
  colors = c("#1A5276", "#76D7C4"),
  point_size = 0.8,
  alpha = 1,
  threshold = NULL,
  threshold_color = "#E74C3C",
  highlight = NULL,
  highlight_color = "#E74C3C",
  highlight_size = 2,
  label_ids = NULL,
  label_top_n = NULL,
  chromosomes = NULL,
  chr_labels = NULL,
  y_label = NULL,
  y_limit = NULL,
  title = NULL
)
```

Arguments

`data`	A data.frame (or tibble/data.table) with one row per feature, or a GenomicRanges::GRanges whose metadata columns hold the value (the range start is used as the position).
`value`	Name of the column holding the value to plot on the y-axis. Required.

`chr`, `bp` Column names for chromosome and genomic position (auto-detected if NULL). Ignored when `annotation` is set, because the coordinates are then looked up from the gene IDs.

`id` Column holding gene identifiers (gene IDs or symbols), used when `annotation` is set (auto-detected if NULL).

`annotation` Optional gene models used to look up coordinates for the `id` column: a GenomicRanges::GRanges, `TxDb` or `EnsDb` object (see `annotate_positions()`). `NULL` (default) means the data already carry `chr`/`bp` columns.

`id_type` Whether `id` holds gene IDs (`"gene_id"`, default) or gene symbols (`"symbol"`).

`label` Column holding feature labels (e.g. gene symbol), used by `highlight`, `label_ids` and `label_top_n`. When `annotation` is set and `label` is NULL, the looked-up gene symbol is used.

`colors` Two-color vector for alternating chromosomes.

`point_size` Point size.

`alpha` Point transparency.

`threshold` Optional numeric vector of y-values at which to draw horizontal reference lines (in the units of `value`).

`threshold_color` Color for the threshold lines.

`highlight` Values of the `label` column to highlight.

`highlight_color` Color for highlighted features.

`highlight_size` Size for highlighted points.

`label_ids` Values of the `label` column to annotate with text.

`label_top_n` Annotate the N features with the largest value (largest absolute value when the data contains negative values).

`chromosomes` Subset of chromosomes to plot (integer vector).

`chr_labels` Custom chromosome labels. Options: NULL (all labels), `"odd"` (only odd-numbered chromosomes labeled), or a character vector of labels (same length as displayed chromosomes).

`y_label` Y-axis title. Defaults to the `value` column name.

`y_limit` Upper y-axis limit.

`title` Plot title.

Value

A ggplot object.

Examples

```
set.seed(1)
df <- data.frame(
  chr = rep(1:5, each = 200),
  bp = as.integer(runif(1000, 1, 2e8)),
  score = round(runif(1000, 0, 1e5)),
  gene = paste0("g", 1:1000)
)

# Basic value Manhattan
value_manhattan(df, value = "score")

# Label the top features and set a custom y-axis title
value_manhattan(df, value = "score", label = "gene",
                label_top_n = 5, y_label = "ranking difference")

# Add a reference line and restrict to a few chromosomes
value_manhattan(df, value = "score", threshold = 90000,
                chromosomes = 1:3)
```

---

volcano_plot *GWAS effect-size volcano plot*

Description

Plot effect size (BETA) against statistical significance (-log10(p)), revealing both the magnitude and direction of genetic effects. Unlike RNA-seq volcano plots, this variant can color by chromosome and scale point size by allele frequency.

Usage

```
volcano_plot(
  data,
  beta = NULL,
  p = NULL,
  snp = NULL,
  p_threshold = 5e-08,
  beta_threshold = NULL,
  color_by = "significance",
  size_by = NULL,
  point_size = 0.8,
  alpha = 0.6,
  label_snps = NULL,
  label_top_n = NULL,
  colors = c(significant = "#E74C3C", nonsignificant = "#BDC3C7"),
  title = NULL
)
```

Arguments

`data`	A `gwas_data` object or data.frame.
`beta`, `p`, `snp`	Column name overrides.
`p_threshold`	P-value significance threshold.
`beta_threshold`	Minimum |BETA| to consider significant.
`color_by`	How to color points: "significance", "chromosome", or a column name.
`size_by`	Column name to scale point size (e.g. "AF"), or NULL.
`point_size`	Base point size (used when size_by is NULL).
`alpha`	Point transparency.
`label_snps`	Character vector of SNP IDs to label.
`label_top_n`	Label the top N most significant SNPs.
`colors`	Named color vector: "significant" and "nonsignificant" for significance mode, or any custom palette.
`title`	Plot title.

Value

A ggplot object.

Examples

```
data(example_gwas, package = "ggwas")

# Basic volcano
volcano_plot(example_gwas)

# Label top hits
volcano_plot(example_gwas, label_top_n = 5)

# Color by chromosome
volcano_plot(example_gwas, color_by = "chromosome", label_top_n = 5)

# Scale point size by allele frequency
volcano_plot(example_gwas, size_by = "AF", label_top_n = 3)
```

Index

∗ **datasets**
example_gwas, 17

annotate_genes, 3
annotate_genes(), *4, 5*, *33*, *54*
annotate_positions, 4
annotate_positions(), *58*
architecture_plot, 5
as_granges, 6
as_gwas_data, 7
as_gwas_data(), *6*

calc_lambda, 8
circular_manhattan, 9
circular_manhattan(), *25*
coloc_plot, 11

data.table::fread(), *8*, *44–48*
density_signal_plot, 12

effect_compare_plot, 13
enrichment_manhattan, 15
example_gwas, 17

filter_region, 18
finemapping_plot, 18
forest_plot, 20

gene_track, 21
gene_track(), *3*, *31*
genetic_correlation, 23
GenomicRanges::distanceToNearest(), *3*
GenomicRanges::GRanges, *3–5*, *11*, *15*, *19*, *21*, *22*, *28*, *31*, *52*, *54*, *57*, *58*
get_loci, 24
ggplot2::scale_color_manual(), *49, 50*
GRanges, *6, 7*
gwas_palette, 25
gwas_palette(), *9*, *16*, *27*, *39*, *50*
gwas_palettes, 25
gwas_preset, 26
gwas_summary, 26
gwas_summary(), *25*

highlight_regions, 28

journal_themes, 29

locus_plot, 30
locus_plot(), *22*

maf_filter, 32
manhattan_genes, 32
manhattan_plot, 33
manhattan_plot(), *25*, *28*, *33*, *57*
merge_gwas, 36
miami_plot, 37
miami_plot(), *25*
multitrait_manhattan, 38

phewas_plot, 40
pvalue_heatmap, 41

qq_plot, 42

read_gcta_mlma, 44
read_gemma, 45
read_gwas_table, 45
read_plink_assoc, 46
read_plink_linear, 47
read_plink_logistic, 47
read_regenie, 48

scale_color_chromosome, 49
scale_color_gwas, 50
scale_color_gwas(), *25*
scale_fill_chromosome *(scale_color_chromosome)*, 49
scale_fill_gwas *(scale_color_gwas)*, 50
set_sex_chr_map, 50

snp_density, 51

theme_cell *(journal_themes)*, 29
theme_gwas, 53
theme_gwas(), *27*
theme_nature *(journal_themes)*, 29
theme_plos *(journal_themes)*, 29
theme_poster *(journal_themes)*, 29
theme_presentation *(journal_themes)*, 29
theme_science *(journal_themes)*, 29
top_hits, 54
trumpet_plot, 55

validate_gwas_data, 56
value_manhattan, 57
value_manhattan(), *4, 5*
volcano_plot, 59