Package: ggwas (via r-universe)

July 23, 2026

Title Modern 'ggplot2' Visualizations for Genome-Wide Association Studies

Version 0.99.5

Description Create publication-ready visualizations for genome-wide association studies. Provides 20 plot types including 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. 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 true

Roxygen list(markdown = TRUE)

Depends R (>= 4.5.0)

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

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

VignetteBuilder knitr

biocViews Visualization, GenomeWideAssociation, Annotation, DataImport, Software

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-07-23 17:01:11 UTC

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

RemoteRef HEAD

RemoteSha 10e028ecaa15759f10b7e0f18d52d4bcee424849

Contents

`annotate_genes` *Annotate peaks with nearest gene names*

Description

Maps significant SNPs to their nearest gene using a provided gene annotation table. Returns the data with a `gene` column, or adds gene labels directly to a Manhattan plot.

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 or data.frame.
`genes`	A data.frame with gene positions. Required columns: `chr` (integer or "chr1" format), `start`, `end`, `gene` (gene symbol). Optional: `strand`.
`p_threshold`	Only annotate SNPs with p < this value.
`max_distance`	Maximum distance (bp) from SNP to gene midpoint to assign annotation. 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

A data.frame with added `nearest_gene` and `gene_distance` columns.

Examples

```
data(example_gwas)
genes <- data.frame(
  chr = c(1, 2, 5), start = c(1e6, 50e6, 80e6),
  end = c(1.5e6, 51e6, 82e6), gene = c("GENE_A", "GENE_B", "GENE_C")
)
annotated <- annotate_genes(example_gwas, genes, top_n = 10)
head(annotated[!is.na(annotated$nearest_gene), ])
```

`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 that will be back-transformed.

Value

A tibble with class `gwas_data`.

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)
```

---

chr_info *Chromosome information for common species*

Description

Built-in chromosome lengths and centromere positions for human, mouse, and cattle genomes. For other species, use `chr_info_ucsc()` to fetch data from UCSC or supply a custom data.frame.

Usage

```
chr_info_human(build = "hg38")

chr_info_mouse(build = "mm39")

chr_info_cattle(build = "ARS-UCD1.2")
```

Arguments

build	Genome build (default: latest).

Value

A data.frame with columns: chr (integer), length, centromere_start, centromere_end.

Examples

```
chr_info_human()
chr_info_mouse()
chr_info_cattle()
```

---

chr_info_ucsc *Fetch chromosome information from UCSC*

Description

Downloads chromosome lengths and centromere positions from the UCSC Genome Browser for any supported genome assembly. Requires internet access.

Usage

```
chr_info_ucsc(genome, autosomes_only = TRUE)
```

Arguments

genome	UCSC genome identifier, e.g. "hg38", "mm39", "bosTau9", "canFam6", "susScr11", "galGal6".
autosomes_only	If TRUE (default), return only autosomal chromosomes (exclude X, Y, M, and unplaced scaffolds).

Value

A data.frame with columns: chr (integer), length, centromere_start, centromere_end.

Examples

```
## Not run:
# Dog genome
dog <- chr_info_ucsc("canFam6")
snp_density(gwas, chr_info = dog)

# Pig genome
pig <- chr_info_ucsc("susScr11")

## End(Not run)
```

`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_chr = NULL,
  region_start = NULL,
  region_end = 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_chr`	Chromosome of the region to plot.
`region_start`, `region_end`	Start and end positions.
`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_chr = 1, region_start = 1e6, region_end = 30e6,
           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,
  chr_info = 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).
`chr_info`	Optional data.frame with `chr` and `length` columns. 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)

# With chromosome info for accurate lengths
density_signal_plot(example_gwas, bin_size = 5e6,
  chr_info = chr_info_human())

# 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 or data.frame.
`annotations`	A data.frame or list of data.frames with annotation regions. Each must have columns: `chr` (integer), `start`, `end`. Optionally: `category` (annotation type), `name` (region label). If a named list, names are used as category labels.
`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, uses a colorblind-friendly 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)
# Define annotation regions
annot <- data.frame(
  chr = c(1, 2, 5),
  start = c(1e6, 50e6, 100e6),
  end = c(5e6, 55e6, 110e6),
  category = c("Enhancer", "Gene", "eQTL")
)
enrichment_manhattan(example_gwas, annotations = annot)
```

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

```
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

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_chr = NULL,
  region_start = NULL,
  region_end = 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_chr`, `region_start`, `region_end`	Region to plot.
`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_chr = 1,
                 region_start = 1e6, region_end = 50e6)
```

`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_annotation *Built-in protein-coding gene annotations*

Description

Return a data.frame of protein-coding gene positions bundled with ggwas, ready for `gene_track()`, `locus_plot()` or `manhattan_genes()`, so that regional and gene-labelled plots work without downloading a GTF. Data are derived from Ensembl (GRCh38 release 103; GRCh37 release 75) and restricted to protein-coding genes.

Usage

```
gene_annotation(build = c("GRCh38", "GRCh37"))
```

Arguments

`build`	Genome build: `"GRCh38"` (default) or `"GRCh37"`.

Value

A data.frame with columns `chr` (integer), `start`, `end`, `gene`, and `strand`.

Examples

```
genes <- gene_annotation("GRCh38")
head(genes)

# Gene track for a region, no GTF needed
gene_track(genes, region_chr = 6, region_start = 25e6, region_end = 34e6)
```

---

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 `exon_data` is supplied, each gene is rendered as an intron backbone with exon boxes and a strand arrow.

Usage

```
gene_track(
  gene_data,
  region_chr,
  region_start,
  region_end,
  exon_data = NULL,
  highlight_genes = NULL,
  highlight_color = "#E74C3C",
  label_size = 2.5,
  track_color = "#1A5276",
  show_strand = TRUE,
  max_genes = 50
)
```

Arguments

`gene_data`	A data.frame with columns: chr, start, end, gene. Optional: strand ("+"/"-").
`region_chr`	Chromosome to display (integer or string).
`region_start`, `region_end`	Region boundaries in base pairs.
`exon_data`	Optional data.frame of exons (columns chr, start, end, gene) matching genes in `gene_data` by the `gene` column. When supplied, genes are drawn with exon structure. Read with `read_gtf(path, feature_type = "exon")`.
`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()`.

Examples

```
genes <- data.frame(
  chr = c(1, 1, 1), start = c(1e6, 5e6, 8e6),
  end = c(2e6, 6e6, 9e6), gene = c("GeneA", "GeneB", "GeneC"),
  strand = c("+", "-", "+")
)
```

```
# Gene-body track
gene_track(genes, region_chr = 1, region_start = 0, region_end = 10e6)

# With exon structure
exons <- data.frame(
  chr = 1,
  start = c(1.0e6, 1.5e6, 5.0e6, 5.6e6, 8.1e6),
  end   = c(1.1e6, 1.7e6, 5.2e6, 5.9e6, 8.4e6),
  gene  = c("GeneA", "GeneA", "GeneB", "GeneB", "GeneC")
)
gene_track(genes, 1, 0, 10e6, exon_data = 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 optimized for GWAS visualizations. All palettes are colorblind-friendly by default.

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

```
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

```
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`	A data.frame with columns: `chr` (integer), `start`, `end`. Optional: `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 <- data.frame(
  chr = c(6, 1), start = c(25e6, 5e6), end = c(35e6, 20e6),
  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_chr = NULL,
  region_start = NULL,
  region_end = 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),
  gene_data = 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_chr`	Chromosome of the region to plot.
`region_start`, `region_end`	Start and end positions.
`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.
`gene_data`	A data.frame with columns: chr, start, end, gene, strand.
`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 patchwork composition if gene_data is provided).

Examples

```
data(example_gwas, package = "ggwas")
locus_plot(example_gwas, region_chr = 1,
           region_start = 1e6, region_end = 20e6)
```

`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 annotation data.frame (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 <- data.frame(
  chr = c(1, 2, 5, 7, 11), start = c(1e6, 50e6, 80e6, 30e6, 60e6),
  end = c(2e6, 52e6, 85e6, 35e6, 65e6),
  gene = c("BRCA1", "FTO", "TCF7L2", "CDKAL1", "KCNQ1")
)
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 to cut out a middle region. Either a single value (break point, resumes at max value) or a vector of two values `c(break_from, resume_at)` defining the cut range in -log10(p) units. For example, `y_truncate = c(15, 50)` shows 0-15 at full scale, cuts 15-50, then shows 50+ above the break.

`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_gtf *Read gene annotations from GTF/GFF3 file*

Description

Parse a GTF or GFF3 annotation file and return a data.frame of gene positions ready for use with gene_track() or locus_plot(). Supports .gz compressed files.

Usage

```
read_gtf(
  path,
  feature_type = "gene",
  gene_name_attr = c("gene_name", "Name", "gene"),
  gene_id_attr = "gene_id",
  biotype = NULL
)
```

Arguments

path	Path to GTF or GFF3 file (plain text or gzipped).
feature_type	Feature type to extract (column 3 of GTF). Default "gene".
gene_name_attr	Attribute key(s) to use as gene name, tried in order. For GTF: "gene_name", for GFF3: "Name".
gene_id_attr	Attribute key for gene ID (e.g. ENSG...).
biotype	Character vector of gene biotypes to keep, e.g. "protein_coding". If NULL (default), all biotypes are returned.

Value

A data.frame with columns: chr (integer), start, end, gene, strand, gene_id. Ready for gene_track() or locus_plot(gene_data = ...).

Examples

```
f <- system.file("extdata", "example.gtf", package = "ggwas")
genes <- read_gtf(f)
head(genes)
```

---

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"), ...)

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

Arguments

colors	Two-element character vector of alternating colors.
...	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 *Colorblind-safe chromosome color scale*

Description

Scale using colorblind-friendly alternating colors for chromosomes.

Usage

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

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

Arguments

palette	Palette name from gwas_palette().
...	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

By default, ggwas uses human chromosome conventions where chromosome 23 is labeled "X", 24 is "Y", etc. For non-human organisms or when chromosomes should be displayed as plain numbers, this mapping can be customized or disabled.

Usage

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

Arguments

`map`	A named integer vector mapping labels to integers, e.g. `c(X = 23L, Y = 24L)`. Pass `NULL` to disable all sex chromosome mapping (chromosomes displayed as plain numbers).

Value

The previous mapping (invisibly).

Examples

```
# Default: human (23 -> X, 24 -> Y, 25 -> XY, 26 -> MT)
set_sex_chr_map()

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

# Mouse: 20 -> X, 21 -> Y
set_sex_chr_map(c(X = 20L, Y = 21L))

# Restore default
set_sex_chr_map()
```

`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 `chr_info` is 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"),
  chr_info = 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.
`chr_info`	Optional data.frame with columns `chr`, `length`, and optionally `centromere_start`, `centromere_end`. Use `chr_info_human()` for hg38. If NULL, chromosome lengths are inferred from the data.
`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 centromere positions are available in `chr_info`, 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)
snp_density(example_gwas, bin_size = 5e6, chr_info = chr_info_human())

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

# Points style: individual variants on chromosome outlines
snp_density(example_gwas, style = "points", chr_info = chr_info_human())

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

# Uniform color, no centromeres
snp_density(example_gwas, style = "points", color_by = "uniform",
  show_centromeres = FALSE)
```

`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 annotation data.frame (chr, start, end, gene). If provided, nearest gene is added.
`max_gene_distance`	Maximum distance to assign a 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)
```

`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