Package: locusviz (via r-universe)

May 31, 2026

Type Package

Title Yet Another Locus Visualization Package in R

Version 0.3.0

Author Masahiro Kanai

Maintainer Masahiro Kanai <mkanai@broadinstitute.org>

Description This package provides various functions to visualize GWAS data for a locus of interest using ggplot2.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.5.0), ggplot2

Imports AnnotationDbi, binom, biovizBase, boot, BuenColors, cowplot, data.table, dplyr, ensembldb, epitools, forcats, GenomeInfoDb, GenomicFeatures, GenomicRanges, ggbio, ggrastr, IRanges, magrittr, patchwork, purrr, rlang, rtracklayer, scales, shades, stringr, tibble, tidyr, Hmisc, jsonlite

Suggests ComplexHeatmap, trackViewer, txdbmaker

Remotes caleblareau/BuenColors

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

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

Date/Publication 2026-05-31 00:49:19 UTC

RemoteUrl https://github.com/mkanai/locusviz

RemoteRef HEAD

RemoteSha 0652daff9ad3d4100d65d30e9183bb24ae3a92ec

Contents

`annotate_hrange` *Annotate horizontal range on a plot*

Description

This function adds a horizontal line annotation with optional tips and label to indicate a range or distance on a plot.

Usage

```
annotate_hrange(
  xmin,
  xmax,
  y,
  scale = 1,
  label = NULL,
  tip_length = 0,
  line.size = 0.2,
  text.size = 2
)
```

Arguments

`xmin`	Numeric value for the start position of the range
`xmax`	Numeric value for the end position of the range
`y`	Numeric value for the y-axis position of the annotation
`scale`	Numeric scaling factor for partial ranges (default: 1). Values < 1 create a broken line to indicate continuation
`label`	Character string to display at the center of the range (optional)
`tip_length`	Numeric length of vertical tips at range endpoints (default: 0)
`line.size`	Numeric width of the annotation lines (default: 0.2)
`text.size`	Numeric size for the label text (default: 2)

Value

A list of ggplot2 layers (geom_segment and optionally geom_text)

Examples

```
## Not run:
ggplot(data, aes(x, y)) +
  geom_point() +
  annotate_hrange(xmin = 100, xmax = 200, y = 5, label = "100 kb")

## End(Not run)
```

`annotate_r2` *Annotate variants with linkage disequilibrium (r2) values*

Description

Annotate variants with linkage disequilibrium (r2) values

Usage

```
annotate_r2(
  df,
  lead_variant = NULL,
  lead_variant_col = "lead_variant",
  reference_panel = c("1000G", "sisu42"),
  window = 5e+05,
  population = "EUR"
)
```

Arguments

`df`	Data frame containing variant information with a 'variant' column
`lead_variant`	Lead variant in chr:pos:ref:alt or chr_pos_ref_alt format. If NULL, will attempt to use the variant marked as TRUE in the column specified by `lead_variant_col`
`lead_variant_col`	Name of the logical column in df that indicates the lead variant (default: "lead_variant"). Used only when `lead_variant` is NULL
`reference_panel`	Reference panel to use ("1000G" or "sisu42")
`window`	Window size around lead variant in base pairs (default: 500000)
`population`	For 1000G panel, the population code (e.g., "EUR", "AFR", "EAS", "SAS", "AMR")

Value

Data frame with r2 values annotated

binom_ci *Binomial confidence interval*

Description

Computes a confidence interval for a binomial proportion.

Usage

```
binom_ci(x, n, methods = "wilson", colname = "frac")
```

Arguments

x	Number of successes
n	Number of trials
methods	Method for CI computation (default: "wilson"). See binom.confint for available methods.
colname	Column name for the output fraction (default: "frac")

Value

A tibble with three columns: the proportion, lower CI bound, and upper CI bound

Examples

```
## Not run:
binom_ci(x = 30, n = 100)

## End(Not run)
```

boot_ci *Bootstrap confidence interval*

Description

Computes a bootstrap confidence interval for a summary statistic.

Usage

```
boot_ci(x, func, conf = 0.95, R = 1000, colname = "boot")
```

Arguments

x	Numeric vector of data values
func	Function to compute the statistic (e.g., mean, median)
conf	Confidence level (default: 0.95)
R	Number of bootstrap replicates (default: 1000)
colname	Column name for the output statistic (default: "median")

Value

A tibble with three columns: the statistic value, lower CI bound, and upper CI bound

---

compute_distance_to_gene

Compute distance from reference position to genes

Description

This function calculates the distance from a reference genomic position to genes in a specified region, using either gene body or TSS distance metrics.

Usage

```
compute_distance_to_gene(
  txdb,
  chromosome,
  start,
  end,
  ref_position,
  type = c("GB", "TSS")
)
```

Arguments

txdb	A TxDb object containing transcript annotations
chromosome	Character string specifying the chromosome (e.g., "chr1" or "1")
start	Numeric start position of the genomic region
end	Numeric end position of the genomic region
ref_position	Numeric reference position from which to calculate distances
type	Character string specifying distance type: "GB" (gene body) or "TSS" (transcription start site)

Value

A data frame with columns: gene (factor ordered by position), method ("Distance" or "Distance_TSS"), and score (numeric distance)

Examples

```
## Not run:
# Calculate distance to gene bodies
distances <- compute_distance_to_gene(
  txdb, "chr1", 1000000, 2000000, 1500000,
  type = "GB"
)

## End(Not run)
```

`compute_functional_enrichment` *Compute functional enrichment of variants by PIP bins*

Description

Calculates enrichment of functional consequences in high vs low posterior inclusion probability (PIP) bins using risk ratio estimates.

Usage

```
compute_functional_enrichment(
  data,
  annot_levels,
  pip_bin_breaks = PIP_BIN_BREAKS,
  consequence_col = "consequence",
  maf_match = FALSE,
  seed = 12345
)
```

Arguments

`data`	A data frame containing variant data with columns for max_pip, consequence (or column specified by consequence_col), and optionally max_maf
`annot_levels`	Character vector of ordered consequence annotation levels
`pip_bin_breaks`	Numeric vector of PIP bin breakpoints (default: c(-Inf, 0.01, 0.1, 0.5, 0.9, 1.0))
`consequence_col`	Character string specifying the column name containing consequence annotations (default: "consequence")
`maf_match`	Logical indicating whether to match variants by minor allele frequency (default: FALSE)
`seed`	Random seed for MAF matching (default: 12345)

Value

A data frame with columns:

`consequence`	Functional consequence category
`enrichment`	Risk ratio estimate
`lower`	Lower confidence interval
`upper`	Upper confidence interval
`n_bottom`	Count in bottom PIP bin
`total_bottom`	Total in bottom PIP bin
`n_top`	Count in top PIP bin
`total_top`	Total in top PIP bin

`distinct_shades` *Generate distinct shades of a base color*

Description

Creates a sequence of n colors with different lightness values based on a base color. The function intelligently adjusts lightness based on whether the base color is dark or light to ensure distinct, usable shades.

Usage

```
distinct_shades(base_color, n = 3)
```

Arguments

`base_color`	Character string specifying a color (any format accepted by the shades package: hex, named colors, etc.)
`n`	Integer number of distinct shades to generate (default: 3)

Details

The function uses the Lab color space for perceptually uniform lightness adjustments. For dark colors ($L < 50$), it generates lighter shades. For light colors, it generates shades in both directions.

Value

Character vector of n color values in hex format

Examples

```
## Not run:
# Generate 3 shades of blue
distinct_shades("blue", n = 3)

# Generate 5 shades of a dark color
distinct_shades("#1f77b4", n = 5)

## End(Not run)
```

---

gencode_txdb *Create a TxDb object from GENCODE annotations*

Description

This function creates a TxDb object from GENCODE annotation files, filtering for canonical transcripts and MANE Select transcripts.

Usage

```
gencode_txdb(
  version = "19",
  genome = c("hg19", "hg38"),
  chrs = paste0("chr", seq_len(22))
)
```

Arguments

version	Character string specifying the GENCODE version (default: '19')
genome	Character string specifying the genome build: 'hg19' or 'hg38'
chrs	Character vector of chromosome names to keep (default: chr1-chr22)

Value

A TxDb object containing filtered GENCODE annotations

Examples

```
## Not run:
# Create TxDb for hg38
txdb_hg38 <- gencode_txdb(genome = "hg38")

# Create TxDb for hg19 with specific chromosomes
txdb_hg19 <- gencode_txdb(genome = "hg19", chrs = c("chr1", "chr2"))

## End(Not run)
```

`get_chromosome_sizes` *Get chromosome sizes and cumulative positions*

Description

This function retrieves chromosome sizes from UCSC genome database and calculates cumulative positions for genome-wide plotting.

Usage

```
get_chromosome_sizes(
  reference_genome,
  chromosomes = paste0("chr", c(seq(22), "X", "Y", "M"))
)
```

Arguments

`reference_genome`	Character string specifying the reference genome: "GRCh37" or "GRCh38"
`chromosomes`	Character vector of chromosome names to include (default: chr1-chr22, chrX, chrY, chrM)

Value

A data frame containing chromosome information with columns: chromosome, seqlengths, genome, start, end, mid

Examples

```
## Not run:
# Get sizes for all default chromosomes in GRCh38
chr_sizes_38 <- get_chromosome_sizes("GRCh38")

# Get sizes for specific chromosomes in GRCh37
chr_sizes_37 <- get_chromosome_sizes("GRCh37", c("chr1", "chr2", "chr3"))

## End(Not run)
```

`get_cs_color_mapping` *Get color mapping for credible sets*

Description

This function creates a color mapping for credible set IDs, with optional prioritization of highlighted credible sets.

Usage

```
get_cs_color_mapping(
  cs_ids,
  highlight_cs_ids = NULL,
  colors = BuenColors::jdb_palette("corona")[setdiff(seq(15), c(8, 15))]
)
```

Arguments

`cs_ids`	Character or numeric vector of credible set IDs
`highlight_cs_ids`	Optional character or numeric vector of credible set IDs to prioritize in color assignment (will receive the first colors)
`colors`	Character vector of colors to use for mapping. Default uses BuenColors corona palette excluding certain values

Value

A named character vector mapping credible set IDs to colors

Examples

```
# Basic usage
cs_colors <- get_cs_color_mapping(c("CS1", "CS2", "CS3"))

# With highlighted credible sets
cs_colors <- get_cs_color_mapping(
  c("CS1", "CS2", "CS3", "CS4"),
  highlight_cs_ids = c("CS2", "CS4")
)
```

`get_default_theme` *Get default ggplot2 theme for locusviz plots*

Description

This function returns a consistent ggplot2 theme used across all locusviz plotting functions. It provides a clean, publication-ready appearance with customizable options for hiding axis elements.

Usage

```
get_default_theme(
  fontsize = 7,
  tag.fontsize = 8,
  title.lines = 1,
  legend.position = c(1, 1),
  legend.justification = c(1, 1),
  hide.xlab = FALSE,
  hide.ylab = FALSE,
  hide.xtext = FALSE,
  hide.ytext = FALSE,
  hide.xtitle = FALSE,
  hide.ytitle = FALSE,
  angle.xtext = NULL
)
```

Arguments

`fontsize`	Numeric font size for all text elements (default: 7)
`tag.fontsize`	Numeric font size for plot tag (default: 8)
`title.lines`	Integer number of lines in the plot title; scales the negative bottom margin so multi-line titles still sit inside the panel (default: 1)
`legend.position`	Numeric vector for legend position (default: c(1,1) = top-right)
`legend.justification`	Numeric vector for legend justification (default: c(1,1))
`hide.xlab`	Logical whether to hide both x-axis text and title (default: FALSE)
`hide.ylab`	Logical whether to hide both y-axis text and title (default: FALSE)
`hide.xtext`	Logical whether to hide x-axis text (default: FALSE)
`hide.ytext`	Logical whether to hide y-axis text (default: FALSE)
`hide.xtitle`	Logical whether to hide x-axis title (default: FALSE)
`hide.ytitle`	Logical whether to hide y-axis title (default: FALSE)
`angle.xtext`	Numeric angle for x-axis text rotation (default: NULL for no rotation)

Value

A ggplot2 theme object

Examples

```
# Get default theme
theme_default <- get_default_theme()

# Hide x-axis elements for stacked plots
theme_no_x <- get_default_theme(hide.xlab = TRUE)

# Larger font size
theme_large <- get_default_theme(fontsize = 12)
```

---

get_global_position *Convert chromosomal position to global genomic position*

Description

This function converts chromosome-specific positions to global genomic positions for genome-wide plotting by adding the cumulative chromosome offset.

Usage

```
get_global_position(chromosome, position, reference_genome)
```

Arguments

chromosome	Character vector of chromosome identifiers
position	Numeric vector of positions within chromosomes
reference_genome	Character string specifying the reference genome: "GRCh37" or "GRCh38"

Value

Numeric vector of global genomic positions

Examples

```
# Convert single position
global_pos <- get_global_position("chr2", 1000000, "GRCh38")

# Convert multiple positions
global_pos <- get_global_position(
  c("chr1", "chr2", "chr3"),
  c(1000000, 2000000, 3000000),
  "GRCh37"
)
```

get_gnomad_colors *Get gnomAD population colors*

Description

This function returns standardized colors for gnomAD populations based on the official gnomAD color scheme. Colors are provided for both lowercase and uppercase population codes.

Usage

```
get_gnomad_colors()
```

Value

A named character vector mapping population codes to hex colors. Includes colors for: afr (African), amr (Latino/American), eas (East Asian), eur/nfe (European/Non-Finnish European), fin (Finnish), sas (South Asian), asj (Ashkenazi Jewish), oth (Other), and additional subpopulations

Examples

```
# Get gnomAD colors
pop_colors <- get_gnomad_colors()

# Use in a plot
scale_color_manual(values = get_gnomad_colors())
```

get_pfam_domains *Get Pfam domain annotations for a gene*

Description

This function retrieves Pfam domain annotations for a specified gene and maps them to genomic coordinates.

Usage

```
get_pfam_domains(
  gene_symbol,
  remove.unknown.domains = TRUE,
  genome_build = c("hg19", "hg38"),
  txdb = NULL,
 pfam = readRDS("~/src/github.com/mkanai/ukbb-finemapping/data/pfam.domains.rds")
)
```

Arguments

gene_symbol	Character string specifying the gene symbol
remove.unknown.domains	Logical indicating whether to remove unknown domains (DUF domains). Default: TRUE
genome_build	Character string specifying the genome build: 'hg19' or 'hg38'
txdb	Optional TxDb object. If NULL, will be loaded based on genome_build
pfam	Data frame containing Pfam domain annotations. Default reads from a specific RDS file path

Value

A data frame containing Pfam domain information with genomic coordinates. Columns include: protein_id, chromosome, start, end, Pfam_ID, Pfam_description

Examples

```
## Not run:
# Get Pfam domains for a gene
domains <- get_pfam_domains("BRCA2", genome_build = "hg38")

# Include unknown domains
all_domains <- get_pfam_domains("BRCA2", remove.unknown.domains = FALSE)

## End(Not run)
```

---

get_tss_gene_body *Extract TSS and gene body information from TxDb*

Description

This function extracts transcription start site (TSS) and gene body information from a TxDb object for specified chromosomes.

Usage

```
get_tss_gene_body(txdb, chromosomes = paste0("chr", c(seq(22), "X")))
```

Arguments

txdb	A TxDb object containing transcript annotations
chromosomes	Character vector of chromosome names to process (default: chr1-chr22, chrX)

Value

A data frame containing transcript information with columns: tx_id, tx_name, chromosome, strand, start, end, tss

highlight_vline *Add vertical highlight lines to a plot*

Description

This function adds dashed vertical lines at specified positions to highlight variants or regions of interest across multiple plot panels.

Usage

```
highlight_vline(highlight_pos, size = 0.5)
```

Arguments

highlight_pos	Numeric vector of x-axis positions to highlight. If NULL, no lines are added
size	Numeric line width (default: 0.5)

Value

A geom_vline ggplot2 layer or NULL if highlight_pos is NULL

Examples

```
## Not run:
# Add highlight lines to a plot
ggplot(df, aes(x = position, y = value)) +
  geom_point() +
  highlight_vline(c(100000, 200000))

## End(Not run)
```

jitter_labels *Jitter labels to avoid overlapping*

Description

This function adjusts label positions to prevent overlapping when plotting multiple labels at similar x-coordinates. It uses the trackViewer package's label adjustment algorithms.

Usage

```
jitter_labels(label.pos, xscale)
```

Arguments

label.pos	Numeric vector or data frame of label positions
xscale	Numeric vector of length 2 specifying the x-axis scale limits

Value

Adjusted label positions with jittering applied to avoid overlaps

Examples

```
## Not run:
# Adjust label positions for a plot
positions <- c(100, 105, 110, 115)
adjusted <- jitter_labels(positions, xscale = c(0, 1000))

## End(Not run)
```

---

liftover_variant *Liftover variant positions between genome builds*

Description

This function converts variant positions between hg19 and hg38 genome builds using UCSC chain files.

Usage

```
liftover_variant(variant, genome_build = c("hg19", "hg38"))
```

Arguments

variant	Character vector of variant identifiers in the format "chromosome:position:ref:alt"
genome_build	Character string specifying the target genome build: "hg19" (lifts from hg38 to hg19) or "hg38" (lifts from hg19 to hg38)

Value

A data frame with columns: variant (original), new_variant, new_chromosome, new_position, new_ref, new_alt. Variants that fail to lift over will have NA values in the new_* columns

Examples

```
## Not run:
# Liftover from hg19 to hg38
lifted <- liftover_variant(c("1:1000000:A:G", "2:2000000:C:T"), "hg38")

# Liftover from hg38 to hg19
lifted <- liftover_variant(c("chr1:1000000:A:G"), "hg19")

## End(Not run)
```

`load_txdb` *Load transcript database for gene annotations*

Description

This function loads a pre-built TxDb object for the specified genome build, or returns a user-provided TxDb object.

Usage

```
load_txdb(genome_build = c("hg19", "hg38"), txdb = NULL)
```

Arguments

`genome_build`	Character string specifying the genome build: "hg19" or "hg38"
`txdb`	Optional TxDb object. If provided, this will be returned instead of loading the default TxDb for the genome build

Value

A TxDb object containing transcript annotations for the specified genome build

Examples

```
## Not run:
# Load default TxDb for hg38
txdb <- load_txdb("hg38")

# Use custom TxDb
custom_txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
txdb <- load_txdb("hg38", txdb = custom_txdb)

## End(Not run)
```

mean_ci *Mean confidence interval via bootstrap*

Description

Computes a bootstrap confidence interval for the mean.

Usage

```
mean_ci(x, conf = 0.95, R = 1000, colname = "mean")
```

Arguments

x	Numeric vector of data values
conf	Confidence level (default: 0.95)
R	Number of bootstrap replicates (default: 1000)
colname	Column name for the output statistic (default: "median")

Value

A tibble with three columns: mean, mean_lower, mean_upper

Examples

```
## Not run:
x <- rnorm(100)
mean_ci(x)

## End(Not run)
```

median_ci *Median confidence interval via bootstrap*

Description

Computes a bootstrap confidence interval for the median.

Usage

```
median_ci(x, conf = 0.95, R = 1000, colname = "median")
```

Arguments

x	Numeric vector of data values
conf	Confidence level (default: 0.95)
R	Number of bootstrap replicates (default: 1000)
colname	Column name for the output statistic (default: "median")

Value

A tibble with three columns: median, median_lower, median_upper

Examples

```
## Not run:
x <- rnorm(100)
median_ci(x)

## End(Not run)
```

---

na_and *NA-safe AND operation*

Description

Performs logical AND operation treating NA values as FALSE.

Usage

```
na_and(...)
```

Arguments

...	Logical vectors to combine with AND

Value

Logical vector with NA-safe AND operation

---

na_or *NA-safe OR operation*

Description

Performs logical OR operation treating NA values as FALSE.

Usage

```
na_or(...)
```

Arguments

...	Logical vectors to combine with OR

Value

Logical vector with NA-safe OR operation

normalize_rank *Normalize scores by rank with exponential decay*

Description

This function converts scores to weights based on their rank, applying an exponential decay factor. Higher ranked scores receive exponentially decreasing weights.

Usage

```
normalize_rank(score, decay = 0.5, ties.method = "min")
```

Arguments

score	Numeric vector of scores to be normalized
decay	Numeric decay factor between 0 and 1 (default: 0.5). Smaller values result in faster decay
ties.method	Character string specifying how ties are handled. Options: "average", "first", "last", "random", "max", "min" (default: "min")

Value

Numeric vector of normalized weights based on rank, with NA values receiving weight of 0

Examples

```
# Normalize a vector of scores
scores <- c(10, 20, 15, NA, 25)
weights <- normalize_rank(scores)

# Use faster decay
weights_fast <- normalize_rank(scores, decay = 0.3)
```

or_else *Return first non-NA value*

Description

This utility function returns the first argument if it's not NA, otherwise returns the second argument.

Usage

```
or_else(a, b)
```

Arguments

a	First value to check
b	Alternative value to return if a is NA

Value

a if not NA, otherwise b

---

or_missing *Conditionally return value or NULL*

Description

This utility function returns a value if a predicate is TRUE, otherwise NULL. Useful for conditionally including ggplot2 layers.

Usage

```
or_missing(predicate, value)
```

Arguments

predicate	Logical value determining whether to return the value
value	Any R object to return if predicate is TRUE

Value

The value if predicate is TRUE, otherwise NULL

---

parse_variant *Parse variant string into components*

Description

This function parses variant identifiers in the format "chromosome:position:ref:alt" into separate columns.

Usage

```
parse_variant(variant, sep = ":")
```

Arguments

variant	Character vector of variant identifiers
sep	Character separator used in variant string (default: ":")

Value

A tibble with columns: chromosome, position (numeric), ref, alt

Examples

```
parse_variant(c("1:1000:A:G", "2:2000:C:T"))
```

`plot_fm_panel` *Create fine-mapping panel*

Description

This function creates a panel showing fine-mapping posterior inclusion probabilities (PIPs) for variants in a genomic region, with optional credible set coloring.

Usage

```
plot_fm_panel(
  data,
  highlight_pos = NULL,
  title = NULL,
  legend_title = "95% CS",
  xlim = NULL,
  ylim = c(0, 1),
  ybreaks = seq(0, 1, by = 0.2),
  point.size = 1.5,
  point.size2 = 3,
  background.layers = NULL,
  rasterize = FALSE,
  rasterize.dpi = 300,
  cs.colors = NULL,
  relevel.cs_id = TRUE
)
```

Arguments

`data`	Data frame containing variant data with columns: position, pip, and optionally cs_id for credible set membership
`highlight_pos`	Numeric vector of positions to highlight with larger diamonds
`title`	Character string for plot title
`legend_title`	Character string for legend title (default: "95% CS")
`xlim`	Numeric vector of length 2 specifying x-axis limits (start, end)
`ylim`	Numeric vector of length 2 specifying y-axis limits (default: c(0,1))
`ybreaks`	Numeric vector specifying y-axis break points

point.size	Numeric size for regular variant points (default: 1.5)
point.size2	Numeric size for highlighted variant points (default: 3)
background.layers	List of additional ggplot2 layers to add as background
rasterize	Logical whether to rasterize the scatter plot (default: FALSE)
rasterize.dpi	Numeric DPI for rasterization (default: 300)
cs.colors	Character vector of colors for credible sets
relevel.cs_id	Logical whether to relevel credible set IDs (default: TRUE)

Value

A ggplot2 object showing the fine-mapping panel

Examples

```
## Not run:
# Basic fine-mapping plot (default theme applied automatically)
plot_fm_panel(finemapping_data)

# Override theme by adding it on top
plot_fm_panel(finemapping_data) + get_default_theme(fontsize = 7)

# With custom settings
plot_fm_panel(
  finemapping_data,
  highlight_pos = c(123456, 789012),
  xlim = c(1000000, 2000000),
  title = "Fine-mapping results",
  cs.colors = c("red", "blue", "green")
)

## End(Not run)
```

---

plot_gene_panel *Create gene track panel*

Description

Builds a gene track panel showing gene annotations for a genomic region. Defaults to a pure-ggplot2 implementation (‘engine = "native"‘) that packs genes into rows so neither gene bodies nor their text labels collide horizontally — a key improvement over ‘ggbio::geom_alignment‘, which places labels at fixed offsets and overlaps unreadably in dense loci. Pass ‘engine = "ggbio"‘ to fall back to the original ‘ggbio::geom_alignment‘ rendering.

Usage

```
plot_gene_panel(
  chromosome,
  start,
  end,
  genome_build = c("hg19", "hg38"),
  txdb = NULL,
  highlight_pos = NULL,
  highlight_pos_y = NULL,
  gene_col = BuenColors::jdb_palette("calma_azules")[6],
  fontsize = 7,
  point.size = 2,
  label.size = 2,
  arrow.rate = 0.015,
  length = unit(0.1, "cm"),
  background.layers = NULL,
  chars_per_panel = 100,
  max_rows = NULL,
  gene_priority = NULL,
  exon.height = 0.3,
  label.color = "gray30",
  label.offset = 0.45,
  engine = c("native", "ggbio")
)
```

Arguments

chromosome	Character string specifying the chromosome (e.g., "chr1" or "1")
start	Numeric start position of the genomic region
end	Numeric end position of the genomic region
genome_build	Character string specifying genome build: 'hg19' or 'hg38'
txdb	Optional TxDb object. If NULL, will be loaded based on genome_build
highlight_pos	Numeric vector of positions to highlight with diamonds
highlight_pos_y	Numeric y-position for highlight markers. If NULL (default), the markers are placed above the top label row (native) or at y=1 (ggbio).
gene_col	Color for gene tracks (default: blue from calma_azules palette)
fontsize	Numeric font size for plot text (default: 7)
point.size	Numeric size for highlight points (default: 2)
label.size	Numeric size for gene labels (default: 2)
arrow.rate	Numeric. Fraction of the panel x-range used as the target spacing between consecutive strand arrowheads on a gene body. Default 0.015 yields roughly one arrowhead per ~1.5 Setting this to 0 disables strand arrowheads.
length	Unit object specifying strand arrowhead size (default: unit(0.1, "cm"))

`background.layers`
: List of additional ggplot2 layers to add as background

`chars_per_panel`
: (native engine only) Approximate number of characters that fit across the panel at the active font size. Lower values give more horizontal padding per label (and hence more rows); raise it if your figure is wider than ~6in. Default: 100.

`max_rows`
: (native engine only) Optional integer cap on the number of gene rows. Genes that don't fit are silently dropped. NULL (default) means no cap.

`gene_priority`
: (native engine only) Optional character vector of gene symbols to pack first. Anything not listed competes for the remaining rows in genomic order. Useful with ‘max_rows‘ to guarantee that specific genes are kept.

`exon.height`
: (native engine only) Numeric vertical extent of exon rectangles in row units (default: 0.3, so exons span row±0.15).

`label.color`
: (native engine only) Character color for gene labels (default: "gray30").

`label.offset`
: (native engine only) Numeric vertical offset of the label above its gene body, in row units (default: 0.45).

`engine`
: Either ‘"native"‘ (default; collision-aware ggplot2 implementation) or ‘"ggbio"‘ (falls back to ‘ggbio::geom_alignment‘ — useful for reproducing prior figures).

Details

Row assignment (native engine) packs each gene's body interval *unioned with* the bounding box of its text label (estimated from ‘chars_per_panel‘). A row can hold multiple genes only when their labels also fit side-by-side; densely labelled regions naturally spill onto more rows instead of stacking labels on top of each other.

Strand direction is shown by repeated open arrowheads spaced along each gene body, with spacing controlled by ‘arrow.rate‘.

Value

A ggplot2 object showing the gene track panel

Examples

```
## Not run:
# Basic gene panel
plot_gene_panel("chr1", 1000000, 2000000)

# Dense locus — naturally grows extra rows so labels never overlap
plot_gene_panel("chr19", 17500000, 19500000, genome_build = "hg38")

# Keep specific genes visible when capping rows
plot_gene_panel(
  "chr19", 17500000, 19500000,
  genome_build = "hg38",
  max_rows = 3,
  gene_priority = c("JUND", "UBA52", "IFI30")
)
```

```
# Fall back to the original ggbio rendering
plot_gene_panel(
  "chr19", 17500000, 19500000,
  genome_build = "hg38",
  engine = "ggbio"
)

## End(Not run)
```

`plot_gene_score_panel` *Create gene score visualization panel*

Description

This function creates a dot plot showing gene scores from various methods, including optional distance-based scores. The plot displays genes on the x-axis and scoring methods on the y-axis, with dot size and opacity representing score magnitude.

Usage

```
plot_gene_score_panel(
  chromosome,
  start,
  end,
  gene_score.data,
  genome_build = c("hg19", "hg38"),
  txdb = NULL,
  highlight_pos = NULL,
  append.distance = TRUE,
  distance.type = c("GB", "TSS"),
  method.levels = NULL,
  colors = NULL,
  fontsize = 7,
  area.max_size = 4
)
```

Arguments

`chromosome`	Character string specifying the chromosome
`start`	Numeric start position of the genomic region
`end`	Numeric end position of the genomic region
`gene_score.data`	Data frame with columns: gene, score, method
`genome_build`	Character string specifying genome build: 'hg19' or 'hg38'
`txdb`	Optional TxDb object. If NULL, will be loaded based on genome_build

highlight_pos	Optional numeric position to highlight for distance calculations
append.distance	Logical whether to append distance-based scores (default: TRUE)
distance.type	Character string specifying distance type: "GB" (gene body) or "TSS" (transcription start site)
method.levels	Character vector specifying the order of scoring methods
colors	Named vector of colors for each method
fontsize	Numeric font size for plot text (default: 7)
area.max_size	Numeric maximum size for dots (default: 4)

Value

A ggplot2 object showing the gene score panel

Examples

```
## Not run:
# Create gene score panel
scores <- data.frame(
  gene = c("GENE1", "GENE2", "GENE3"),
  score = c(0.8, 0.6, 0.9),
  method = "MAGMA"
)
plot_gene_score_panel("chr1", 1000000, 2000000, scores)

## End(Not run)
```

---

plot_locuszoom *Create LocusZoom-style visualization*

Description

This is the main function for creating LocusZoom-style plots. It combines multiple panels (Manhattan plot, fine-mapping, r2/LD, gene track, and gene scores) into a single comprehensive visualization of a genomic locus.

Usage

```
plot_locuszoom(
  data,
  highlight_pos = NULL,
  window = NULL,
  xlim = NULL,
  manhattan.args = list(),
  manhattan.title = NULL,
  manhattan.breaks = ggplot2::waiver(),
```

```
  manhattan.loglog_p = TRUE,
  nlog10p_threshold = 0,
  fm.args = list(),
  fm.ylim = c(0, 1),
  fm.breaks = seq(0, 1, by = 0.2),
  fm.legend_title = "95% CS",
  r2.args = list(),
  gene.args = list(),
  gene_score.args = list(),
  plot.manhattan = TRUE,
  plot.fm = TRUE,
  plot.r2 = FALSE,
  plot.gene = TRUE,
  plot.gene_score = FALSE,
  fontsize = 7,
  ggtheme = NULL,
  patchwork = TRUE,
  rasterize = FALSE,
  rasterize.dpi = 300
)
```

Arguments

`data`	Data frame containing variant information with required columns: chromosome, position, and additional columns depending on enabled panels
`highlight_pos`	Numeric position to highlight across all panels
`window`	Numeric window size around lead variant (ignored if xlim is provided)
`xlim`	Numeric vector of length 2 specifying the x-axis limits (start, end)
`manhattan.args`	List of additional arguments passed to plot_manhattan_panel
`manhattan.title`	Character string for Manhattan panel title
`manhattan.breaks`	Y-axis breaks for Manhattan panel (default: automatic)
`manhattan.loglog_p`	Logical whether to use log-log p-value transformation
`nlog10p_threshold`	Numeric threshold for -log10(p) values
`fm.args`	List of additional arguments passed to plot_fm_panel
`fm.ylim`	Numeric vector for fine-mapping panel y-axis limits (default: c(0,1))
`fm.breaks`	Numeric vector for fine-mapping panel y-axis breaks
`fm.legend_title`	Character string for fine-mapping legend title
`r2.args`	List of additional arguments passed to plot_r2_panel
`gene.args`	List of additional arguments passed to plot_gene_panel

gene_score.args	List of additional arguments passed to plot_gene_score_panel
plot.manhattan	Logical whether to include Manhattan panel (default: TRUE)
plot.fm	Logical whether to include fine-mapping panel (default: TRUE)
plot.r2	Logical whether to include r2/LD panel (default: FALSE)
plot.gene	Logical whether to include gene track panel (default: TRUE)
plot.gene_score	Logical whether to include gene score panel (default: FALSE)
fontsize	Numeric font size for all panels (default: 7)
ggtheme	Optional ggplot2 theme applied on top of every panel's default theme. Use to override styling uniformly across all panels (e.g. 'theme(legend.position = "bottom")'). Default: NULL (no override).
patchwork	Logical whether to combine panels using patchwork (default: TRUE)
rasterize	Logical whether to rasterize scatter plots (default: FALSE)
rasterize.dpi	Numeric DPI for rasterization (default: 300)

Value

Either a combined patchwork plot (if patchwork=TRUE) or a list of individual ggplot2 objects for each panel

Examples

```
## Not run:
# Basic LocusZoom plot
plot_locuszoom(gwas_data, highlight_pos = 123456789)

# Custom configuration with specific panels
plot_locuszoom(
  gwas_data,
  window = 500000,
  plot.r2 = TRUE,
  plot.gene_score = TRUE,
  manhattan.title = "GWAS results for trait X"
)

## End(Not run)
```

---

plot_lollipop *Create lollipop plot for variant effects*

Description

This function creates a lollipop plot showing variant effects on a gene, with positive and negative effects displayed above and below the gene track. It can also display Pfam domains and ClinVar annotations.

Usage

```
plot_lollipop(
  df,
  gene_symbol,
  point_colors,
  clinvar,
  point_shapes = cohort_shapes,
  trait_idx = NULL,
  gene_col = "grey90",
  color_by_cohort = FALSE,
  plot.domains = TRUE,
  remove.unknown.domains = TRUE,
  omit_spacer = FALSE,
  extend.size = NULL,
  plot.extra.genes = FALSE,
  genome_build = c("hg19", "hg38"),
  txdb = NULL
)
```

Arguments

df	Data frame containing variant data with columns: position, pip, susie.beta_posterior, trait, cohort, label
gene_symbol	Character string or vector of gene symbols to plot
point_colors	Named vector of colors for traits or cohorts
clinvar	Data frame containing ClinVar annotations
point_shapes	Named vector of shapes for cohorts (default: cohort_shapes)
trait_idx	Data frame mapping traits to indices (auto-generated if NULL)
gene_col	Color for gene track (default: 'grey90')
color_by_cohort	Logical whether to color by cohort instead of trait
plot.domains	Logical whether to plot Pfam domains (default: TRUE)
remove.unknown.domains	Logical whether to remove unknown domains (default: TRUE)
omit_spacer	Logical whether to omit empty panels (default: FALSE)
extend.size	Numeric or length-2 vector to extend plot region beyond gene
plot.extra.genes	Logical whether to plot additional genes in region
genome_build	Character string specifying genome build: 'hg19' or 'hg38'
txdb	Optional TxDb object. If NULL, will be loaded based on genome_build

Value

A patchwork combined plot object showing the lollipop visualization

Examples

```
## Not run:
# Create lollipop plot for a gene
plot_lollipop(variant_df, "BRCA2", trait_colors, clinvar_data)

## End(Not run)
```

`plot_manhattan_panel` *Create Manhattan plot panel*

Description

This function creates a Manhattan plot panel showing GWAS p-values for a genomic region, with optional LD coloring relative to a lead variant.

Usage

```
plot_manhattan_panel(
  data,
  highlight_pos = NULL,
  xlim = NULL,
  ylim = NULL,
  ybreaks = ggplot2::waiver(),
  nlog10p_threshold = 1,
  loglog_p = 10,
  plot.loglog_p = FALSE,
  point.size = 1.5,
  point.size2 = 3,
  line.size = 0.5,
  title = NULL,
  r2_cols = c("navy", "lightskyblue", "green", "orange", "red"),
  lead_variant_col = "purple3",
  background.layers = NULL,
  rasterize = FALSE,
  rasterize.dpi = 300
)
```

Arguments

`data`	Data frame containing variant data with columns: chromosome, position, nlog10p, lead_variant (logical), and optionally r2
`highlight_pos`	Numeric vector of positions to highlight with larger diamonds
`xlim`	Numeric vector of length 2 specifying x-axis limits (start, end)
`ylim`	Numeric vector of length 2 specifying y-axis limits
`ybreaks`	Numeric vector specifying y-axis break points

`nlog10p_threshold`
: Numeric minimum -log10(p) value to display (default: 1)

`loglog_p`
: Numeric threshold for log-log transformation (default: 10)

`plot.loglog_p`
: Logical whether to use log-log p-value transformation

`point.size`
: Numeric size for regular variant points (default: 1.5)

`point.size2`
: Numeric size for highlighted/lead variant points (default: 3)

`line.size`
: Numeric size for genome-wide significance line (default: 0.5)

`title`
: Character string for plot title

`r2_cols`
: Character vector of colors for r² bins

`lead_variant_col`
: Character color for lead variant (default: "purple3")

`background.layers`
: List of additional ggplot2 layers to add as background

`rasterize`
: Logical whether to rasterize the scatter plot (default: FALSE)

`rasterize.dpi`
: Numeric DPI for rasterization (default: 300)

Value

A ggplot2 object showing the Manhattan plot panel

Examples

```
## Not run:
# Basic Manhattan plot (default theme applied automatically)
plot_manhattan_panel(gwas_data)

# Override theme by adding it on top
plot_manhattan_panel(gwas_data) + get_default_theme(fontsize = 7)

# With custom settings
plot_manhattan_panel(
  gwas_data,
  highlight_pos = c(123456, 789012),
  xlim = c(1000000, 2000000),
  plot.loglog_p = TRUE,
  title = "GWAS results for trait X"
)

## End(Not run)
```

plot_r2_panel *Create r² (linkage disequilibrium) panel*

Description

This function creates a panel showing linkage disequilibrium (r²) values between variants and a lead variant, stratified by population from gnomAD data.

Usage

```
plot_r2_panel(
  data,
  highlight_pos = NULL,
  xlim = NULL,
  ylim = c(0, 1),
  ybreaks = seq(0, 1, by = 0.2),
  point.size = 1.5,
  point.size2 = 3,
  legend.ncol = 2,
  nlog10p_threshold = 1,
  background.layers = NULL,
  rasterize = FALSE,
  rasterize.dpi = 300
)
```

Arguments

data	Data frame containing variant data with columns: variant, position, nlog10p, lead_variant, cs_id, and gnomad_lead_r2_* or gnomad_lead_r_* columns
highlight_pos	Numeric vector of positions to highlight with larger diamonds
xlim	Numeric vector of length 2 specifying x-axis limits (start, end)
ylim	Numeric vector of length 2 specifying y-axis limits (default: c(0,1))
ybreaks	Numeric vector specifying y-axis break points
point.size	Numeric size for regular variant points (default: 1.5)
point.size2	Numeric size for highlighted variant points (default: 3)
legend.ncol	Number of columns for the legend (default: 2)
nlog10p_threshold	Numeric minimum -log10(p) value to display (default: 1)
background.layers	List of additional ggplot2 layers to add as background
rasterize	Logical whether to rasterize the scatter plot (default: FALSE)
rasterize.dpi	Numeric DPI for rasterization (default: 300)

Value

A ggplot2 object showing the r² panel

Examples

```
## Not run:
# Basic r² plot (default theme applied automatically)
plot_r2_panel(gwas_data)

# Override theme by adding it on top
plot_r2_panel(gwas_data) + get_default_theme(fontsize = 7)

# With custom settings
plot_r2_panel(
  gwas_data,
  highlight_pos = c(123456, 789012),
  xlim = c(1000000, 2000000),
  legend.ncol = 3
)

## End(Not run)
```

---

plot_upset *Create UpSet plot for set intersections*

Description

Creates an UpSet-style plot showing set intersections between groups. The plot consists of two panels: a bar plot showing intersection sizes and a matrix plot showing which sets are included in each intersection.

Usage

```
plot_upset(
  df,
  item_col = "item",
  set_col = "set",
  set_colors = NULL,
  degree_colors = NULL,
  base_theme = get_default_theme(),
  log10_scale = FALSE,
  return_list = FALSE
)
```

Arguments

df	A data frame with at least two columns: one for items and one for sets/groups
item_col	Character. Name of the column containing items. Default: ‘"item"‘
set_col	Character. Name of the column containing sets/groups. Default: ‘"set"‘
set_colors	Named vector of colors for each set. If NULL (default), uses default ggplot2 colors generated by ‘scales::hue_pal()‘.
degree_colors	Vector of colors for different intersection degrees (sizes). If NULL (default), uses default ggplot2 discrete fill scale, which automatically adapts to any number of intersection degrees.
base_theme	ggplot2 theme object for the bar plot. Default: ‘get_default_theme()‘
log10_scale	Logical. If TRUE, uses log10 scale for the Y-axis in the bar plot. Default: FALSE
return_list	Logical. If TRUE, returns a list of two ggplot objects (matrix and bar). If FALSE, returns a combined plot using patchwork. Default: FALSE

Value

Either a combined patchwork plot (if ‘return_list = FALSE‘) or a list containing two ggplot objects: the matrix plot and the bar plot

Examples

```
## Not run:
# Create example data
df <- data.frame(
  item = c("A", "B", "C", "D", "E", "F"),
  set = c("X", "X", "Y", "Y", "Z", "Z")
)
plot_upset(df)

# With custom set colors
plot_upset(df, set_colors = c(X = "red", Y = "blue", Z = "green"))

# With custom degree colors (for intersection sizes)
plot_upset(df, degree_colors = c("lightblue", "steelblue", "darkblue"))

# With log10 scale for Y-axis
plot_upset(df, log10_scale = TRUE)

# Return individual plots
plots <- plot_upset(df, return_list = TRUE)

## End(Not run)
```

plot_upset_bar *Plot UpSet bar panel*

Description

Creates the bar chart showing the size of each set intersection

Usage

```
plot_upset_bar(
  matrix_data,
  degree_colors = NULL,
  base_theme,
  log10_scale = FALSE
)
```

Arguments

matrix_data	List containing ‘matrix‘ data frame from ‘prepare_upset_matrix()‘
degree_colors	Vector of colors for different intersection degrees. If NULL (default), uses ggplot2’s default discrete fill scale.
base_theme	ggplot2 theme object
log10_scale	Logical. If TRUE, uses log10 scale for the Y-axis. Default: FALSE

Value

A ggplot2 object

plot_upset_matrix *Plot UpSet matrix panel*

Description

Creates the matrix visualization showing which sets are included in each intersection

Usage

```
plot_upset_matrix(matrix_data, set_colors, base_theme = get_default_theme())
```

Arguments

matrix_data	List containing ‘matrix‘ and ‘segment‘ data frames from ‘prepare_upset_matrix()‘
set_colors	Named vector of colors for each set
base_theme	ggplot2 theme used to derive label text size (default: ‘get_default_theme()‘)

Value

A ggplot2 object

`preprocess` *Preprocess GWAS data for visualization*

Description

This function preprocesses GWAS summary statistics data for use with locusviz plotting functions. It standardizes column names, calculates -log10(p) values, and identifies the lead variant.

Usage

```
preprocess(
  data,
  lead_variant = NULL,
  chromosome_col = "chromosome",
  position_col = "position",
  variant_col = "variant",
  beta_col = "beta",
  se_col = "se",
  pvalue_col = "pvalue",
  pip_col = "pip",
  cs_id_col = "cs_id",
  r2_col = "r2"
)
```

Arguments

`data`	Data frame containing GWAS summary statistics
`lead_variant`	Character string specifying the lead variant ID. If NULL, the variant with the highest -log10(p) value will be selected
`chromosome_col`	Name of the chromosome column (default: "chromosome")
`position_col`	Name of the position column (default: "position")
`variant_col`	Name of the variant ID column (default: "variant")
`beta_col`	Name of the beta/effect size column (default: "beta")
`se_col`	Name of the standard error column (default: "se")
`pvalue_col`	Name of the p-value column (default: "pvalue")
`pip_col`	Name of the PIP column for fine-mapping (default: "pip")
`cs_id_col`	Name of the credible set ID column (default: "cs_id")
`r2_col`	Name of the r^2 column (default: "r2")

Value

A standardized data frame with columns: chromosome, position, variant, beta, se, pip, cs_id, nlog10p, lead_variant, and optionally pvalue and r2

Examples

```
## Not run:
# Basic preprocessing
processed_data <- preprocess(gwas_data)

# With custom column names
processed_data <- preprocess(
  gwas_data,
  chromosome_col = "chr",
  position_col = "pos",
  lead_variant = "rs123456"
)

## End(Not run)
```

`scale_color_chromosome` *Create chromosome color scale for Manhattan plots*

Description

This function creates a ggplot2 color scale that alternates colors between odd and even chromosomes for better visualization in Manhattan plots.

Usage

```
scale_color_chromosome(
  odd_color = "darkblue",
  even_color = "grey50",
  reference_genome = "GRCh37"
)
```

Arguments

`odd_color`	Character string specifying the color for odd-numbered chromosomes (default: "darkblue")
`even_color`	Character string specifying the color for even-numbered chromosomes (default: "grey50")
`reference_genome`	Character string specifying the reference genome: "GRCh37" or "GRCh38" (default: "GRCh37")

Value

A ggplot2 scale_color_manual object

Examples

```
## Not run:
# Default alternating colors
ggplot(df, aes(x = position, y = -log10(p), color = chromosome)) +
  geom_point() +
  scale_color_chromosome()

# Custom colors
ggplot(df, aes(x = position, y = -log10(p), color = chromosome)) +
  geom_point() +
  scale_color_chromosome(odd_color = "red", even_color = "blue")

## End(Not run)
```

`scale_x_chromosome` *Create chromosome x-axis scale for genome-wide plots*

Description

This function creates a ggplot2 x-axis scale for genome-wide plots with chromosome labels at the center of each chromosome and minor breaks at chromosome boundaries.

Usage

```
scale_x_chromosome(reference_genome, ...)
```

Arguments

`reference_genome`	Character string specifying the reference genome: "GRCh37" or "GRCh38"
`...`	Additional arguments passed to scale_x_continuous

Value

A ggplot2 scale_x_continuous object with chromosome-specific breaks and labels

Examples

```
## Not run:
# Create a Manhattan plot with chromosome scale
ggplot(gwas_data, aes(x = global_position, y = -log10(p))) +
  geom_point() +
  scale_x_chromosome("GRCh38")
```

```
# With custom expansion
ggplot(gwas_data, aes(x = global_position, y = -log10(p))) +
  geom_point() +
  scale_x_chromosome("GRCh37", expand = expansion(mult = 0.02))

## End(Not run)
```

`spearman_ci` *Spearman correlation confidence interval*

Description

Computes Spearman's rank correlation with a confidence interval using Fisher's z-transform on the ranks (i.e., Pearson correlation of ranks).

Usage

```
spearman_ci(x, y, conf = 0.95, colname = "rho")
```

Arguments

`x`	Numeric vector
`y`	Numeric vector of the same length as `x`
`conf`	Confidence level (default: 0.95)
`colname`	Column name prefix for the output (default: "rho")

Value

A tibble with four columns: rho, rho_lower, rho_upper, rho_p

Examples

```
## Not run:
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
spearman_ci(x, y)

## End(Not run)
```

stat_summary_irq *Summary statistic with interquartile range*

Description

This function creates a ggplot2 stat_summary layer that displays the median with error bars showing the interquartile range (25th to 75th percentile).

Usage

```
stat_summary_irq(color = "black", size = 0.1)
```

Arguments

color	Character string specifying the color for the summary statistics (default: "black")
size	Numeric size for the lines (default: 0.1)

Value

A ggplot2 stat_summary layer

Examples

```
## Not run:
# Add IQR summary to a plot
ggplot(df, aes(x = group, y = value)) +
  geom_point() +
  stat_summary_irq(color = "red")

## End(Not run)
```

trans_loglog_p *Create log-log transformation for p-values*

Description

This function creates a custom transformation for p-values that applies logarithmic scaling above a threshold to better visualize extreme p-values in Manhattan plots.

Usage

```
trans_loglog_p(loglog_p = 10)
```

Arguments

loglog_p	Numeric threshold above which to apply log-log transformation (default: 10). Values below this threshold are unchanged, values above are log-transformed

Value

A scales transformation object for use with ggplot2 scale functions

Examples

```
## Not run:
# Use in a Manhattan plot
ggplot(gwas_data, aes(x = position, y = -log10(p))) +
  geom_point() +
  scale_y_continuous(trans = trans_loglog_p(10))

# With higher threshold
ggplot(gwas_data, aes(x = position, y = -log10(p))) +
  geom_point() +
  scale_y_continuous(trans = trans_loglog_p(20))

## End(Not run)
```

---

tss_v19_hg19 *TSS and gene body data for GENCODE v19 (hg19/GRCh37)*

Description

Transcription start site and gene body information extracted from GENCODE v19 annotations for the GRCh37/hg19 genome build.

Usage

```
tss_v19_hg19
```

Format

A data frame with columns:

tx_id Transcript ID

tx_name Transcript/gene name

chromosome Chromosome (without 'chr' prefix)

strand Strand (+ or -)

start Transcript start position

end Transcript end position

tss Transcription start site position

tss_v34_hg38 *TSS and gene body data for GENCODE v34 (hg38/GRCh38)*

Description

Transcription start site and gene body information extracted from GENCODE v34 annotations for the GRCh38/hg38 genome build.

Usage

```
tss_v34_hg38
```

Format

A data frame with columns:

tx_id Transcript ID

tx_name Transcript/gene name

chromosome Chromosome

strand Strand (+ or -)

start Transcript start position

end Transcript end position

tss Transcription start site position

tss_v39_hg38 *TSS and gene body data for GENCODE v39 (hg38/GRCh38)*

Description

Transcription start site and gene body information extracted from GENCODE v39 annotations for the GRCh38/hg38 genome build. GENCODE v39 matches the gene models used by gnomAD v4.1.1 (VEP v105).

Usage

```
tss_v39_hg38
```

Format

A data frame with columns:

tx_id Transcript ID

tx_name Transcript/gene name

chromosome Chromosome

strand Strand (+ or -)

start Transcript start position

end Transcript end position

tss Transcription start site position

UpSet2 *Modified UpSet plot*

Description

This function creates an UpSet plot with enhanced visual customization options. UpSet plots are used to visualize intersections of multiple sets.

Usage

```
UpSet2(
  m,
  set_on_rows = TRUE,
  comb_col = "black",
  pt_size = grid::unit(3, "mm"),
  lwd = 2,
  bg_col = "#F0F0F0",
  bg_pt_col = "#CCCCCC",
  set_order = NULL,
  comb_order = NULL,
  top_annotation = NULL,
  right_annotation = NULL,
  row_names_side = "left",
  remove_lines = FALSE,
  ...
)
```

Arguments

m	A matrix or data frame where rows/columns represent sets and values indicate set membership (1) or not (0)
set_on_rows	Logical indicating if sets are on rows (default: TRUE) or columns
comb_col	Character color or vector of colors for combination markers

pt_size	Grid unit object specifying point size (default: unit(3, "mm"))
lwd	Numeric line width for connections (default: 2)
bg_col	Character color for background (default: "#F0F0F0")
bg_pt_col	Character color for background points (default: "#CCCCCC")
set_order	Character vector specifying order of sets
comb_order	Character vector specifying order of combinations
top_annotation	HeatmapAnnotation object for top annotation
right_annotation	HeatmapAnnotation object for right annotation
row_names_side	Character string: "left" or "right" for row name position
remove_lines	Logical whether to remove connecting lines (default: FALSE)
...	Additional arguments passed to ComplexHeatmap::UpSet

Value

An UpSet plot object from ComplexHeatmap

Examples

```
## Not run:
# Create sample data
m <- matrix(c(1, 0, 1, 1, 0, 0, 1, 1, 0), nrow = 3)
rownames(m) <- c("Set1", "Set2", "Set3")

# Basic UpSet plot
UpSet2(m)

# Customized UpSet plot
UpSet2(m, comb_col = "red", pt_size = unit(5, "mm"))

## End(Not run)
```

---

variant_str *Create variant string from components*

Description

This function creates variant identifiers in the format "chromosome:position:ref:alt" from separate components.

Usage

```
variant_str(chromosome, position, ref, alt)
```

Arguments

chromosome	Character vector of chromosome identifiers
position	Numeric vector of positions
ref	Character vector of reference alleles
alt	Character vector of alternative alleles

Value

Character vector of variant identifiers

Examples

```
variant_str("1", 1000, "A", "G")
```

---

variant_str2 *Create variant string from locus and alleles*

Description

This function creates variant identifiers from locus (chromosome:position) and alleles string, handling various formatting.

Usage

```
variant_str2(locus, alleles)
```

Arguments

locus	Character vector of locus strings (chromosome:position)
alleles	Character vector of allele strings (may contain brackets/quotes)

Value

Character vector of variant identifiers

Examples

```
variant_str2("1:1000", "[\"A\",\"G\"]")
```

`write_txdb_files` *Write TxDb and TSS data files*

Description

This function generates and saves TxDb SQLite files and TSS RData files for both hg19 and hg38 genome builds. This is an internal function used to prepare the package data files.

Usage

```
write_txdb_files(chromosomes = paste0("chr", c(seq(22), "X")))
```

Arguments

`chromosomes`	Character vector of chromosome names to include (default: chr1-chr22, chrX)

Value

NULL (invisibly). Files are written to inst/extdata/ and data/ directories

Examples

```
## Not run:
# Generate all data files
write_txdb_files()

## End(Not run)
```

Index