Package: splicelogic (via r-universe)

November 3, 2026

Title splicelogic: differential transcripts to splice events

Version 1.1.5

Description Translate differential transcript usage results into discrete splice events.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://github.com/thelovelab/splicelogic, https://thelovelab.github.io/splicelogic

biocViews AlternativeSplicing, DifferentialSplicing, Transcriptomics, RNASeq, LongRead, Annotation, FunctionalGenomics

BugReports https://github.com/thelovelab/splicelogic/issues

Depends R (>= 4.5.0), plyranges

Imports dplyr, magrittr, GenomicRanges, tibble, IRanges, S4Vectors, rlang, methods, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0), readr, wiggleplotr, GenomicFeatures, AnnotationHub, ggplot2, AnnotationDbi, Seqinfo, BSgenome, Biostrings, BSgenome.Hsapiens.UCSC.hg38

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/pak/sysreqs make libbz2-dev liblzma-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

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

Date/Publication 2026-11-02 22:47:09 UTC

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

RemoteRef HEAD

RemoteSha c25b4e6c28280c939a9f7d0295971ef4298ae1a4

Contents

splicelogic-package . . . 2
create_mock_data . . . 3
event_palette . . . 4
find_events . . . 4
generate_events . . . 8
get_seq . . . 10
plot_transcript_lfc . . . 11
plot_transcripts . . . 13
plot_transcripts_by_gene . . . 15
plot_transcripts_by_pair . . . 16
prepare_exons . . . 17
prepare_exons_by_partition . . . 18
preprocess . . . 20

Index **21**

`splicelogic-package` *splicelogic: differential transcripts to splice events*

Description

For more details on the features of splicelogic, read the vignette: `browseVignettes(package = "splicelogic")`

Author(s)

Maintainer: Beatriz Campillo <beatrizcampillo29@gmail.com> (ORCID)

Authors:

- Beatriz Campillo <beatrizcampillo29@gmail.com> (ORCID)
- Michael Love <michaelisaiahlove@gmail.com> (ORCID)

Other contributors:

- NIH NHGRI [funder]
- Wellcome Trust [funder]

See Also

Useful links:

- https://github.com/thelovelab/splicelogic
- https://thelovelab.github.io/splicelogic
- Report bugs at https://github.com/thelovelab/splicelogic/issues

`create_mock_data` *Create mock GRanges data for splicing event testing*

Description

Create mock GRanges data for splicing event testing

Usage

```
create_mock_data(
  n_genes = 1,
  n_tx_per_gene = 2,
  n_exons_per_tx = 5,
  coef_range = c(-1, 1),
  strand = c("+", "-", "*")
)
```

Arguments

`n_genes`	Number of genes to simulate
`n_tx_per_gene`	Number of transcripts per gene. Use 2 or more: the first transcript of each gene is given a negative estimate and the second a positive one, so a single transcript per gene leaves the set all-negative and the `generate_*()` helpers with nothing to modify.
`n_exons_per_tx`	Number of exons per transcript
`coef_range`	Range of coefficient values to sample from
`strand`	Strand to place every transcript on: `"+"` (default), `"-"`, or `"*"`. On `"-"` the exon ranks are reversed, so exon_rank 1 is the rightmost exon in genomic coordinates.

Value

A GRanges object with simulated transcripts and exons

Examples

```
# create mock data with 2 genes, 4 transcripts
# per gene, and 4 exons per transcript
gr <- create_mock_data(n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4)

# the same, on the minus strand
gr_minus <- create_mock_data(n_genes = 2, n_exons_per_tx = 4, strand = "-")
```

event_palette *Default fill colours for splicelogic event types*

Description

One pastel colour per `event_type` returned by the `find_*()` functions, chosen together with the soft blue and red `plot_transcripts()` uses for the up- and down-regulated transcripts, so the eleven stay as far apart as pastels allow, including under protanopia and deuteranopia. Pastels leave less room between colours than saturated ones, which is why event exons are also outlined and labelled with their type.

Usage

```
event_palette()
```

Value

A character vector of colours, named by event type.

Examples

```
event_palette()
```

find_events *Find splice events from annotated exons*

Description

Functions to find different types of alternative splicing events from preprocessed GRanges exon data. Events include skipped exon (se), included exon (ie), mutatualy exclusive exons (mxe), retained intron (ri), spliced intron (si), alternative 5' and 3' splice sites (a5ss / a3ss), and alternative transcription start and end sites (atss / ates).

`find_ie()` and `find_si()` are the reverse of `find_se()` and `find_ri()`: the same comparison with the up and down transcripts swapped. A spliced intron is an intron of an up transcript that a down transcript retains, i.e. an intron that the up transcript splices out.

`find_a5ss()` / `find_a3ss()` do not require either boundary of an exon to be shared with its partner, so an exon that moved both boundaries is reported as an a5ss *and* an a3ss. A first exon has no acceptor and a last exon has no donor, so those boundaries never produce an a3ss / a5ss; `find_atss()` / `find_ates()` report them instead by comparing where each transcript begins and ends. `exon_rank` runs in transcription order, so this holds on both strands.

When an exon overlaps two or more exons of its partner (a retained or spliced intron), the boundaries facing the intron are left to `find_ri()` / `find_si()`; only the outer boundaries of the group are compared. An exon that retains an intron *and* uses a different outer splice site is therefore reported as both an ri (or si) and an a5ss / a3ss.

Usage

```
find_se(gr, type = c("over", "boundary", "identical"), inverse = FALSE, ...)

find_skipped_exons(
  gr,
  type = c("over", "boundary", "identical"),
  inverse = FALSE,
  ...
)

find_ie(gr, type = c("over", "boundary", "identical"), ...)

find_included_exons(gr, type = c("over", "boundary", "identical"), ...)

find_mxe(gr, type = c("over", "boundary", "identical"), ...)

find_mutually_exclusive_exons(
  gr,
  type = c("over", "boundary", "identical"),
  ...
)

find_ri(gr, inverse = FALSE)

find_retained_introns(gr, inverse = FALSE)

find_si(gr)

find_spliced_introns(gr)

find_a5ss(gr)

find_alternative_5_prime_splice_sites(gr)

find_a3ss(gr)

find_alternative_3_prime_splice_sites(gr)

find_atss(gr)

find_alternative_transcription_start_sites(gr)

find_ates(gr)

find_alternative_transcription_end_sites(gr)

find_all_events(
  gr,
```

```
  type = c("over", "boundary", "identical"),
  verbose = TRUE,
  ...
)
```

Arguments

gr	A GRanges object with exon annotations, including 'tx_id', 'exon', and 'coef_col' metadata columns and preprocessed with preprocess().
type	How an exon flanking a candidate is matched to an exon of the partner transcript. One of: `"over"` (default) any overlap, with no coordinate in common required. The most permissive setting. `"boundary"` the two exons overlap *and* share the inner boundary of the flank, the splice site facing the candidate: the donor of the upstream flank and the acceptor of the downstream one. On `"+"` that is the end of the left exon and the start of the right one; on `"-"` the reverse. The outer boundaries may differ. `"identical"` the two exons are identical (same start and same end). The strictest setting.
inverse	If TRUE, swaps the up and down transcripts: `find_se()` then identifies included exons instead of skipped exons, and `find_ri()` spliced introns instead of retained introns.
...	Further arguments passed to `GenomicRanges::findOverlaps()` when matching flanking exons, e.g. `minoverlap = 5L` to require at least 5 bp of overlap. Do not pass `select`, which changes the return type. `maxgap` can also be used, to let `"over"` and `"boundary"` match exons that do not overlap at all.
verbose	If TRUE, prints progress messages. Default TRUE.

Value

A GRanges object with the detected exon ranges and the following additional metadata columns:

`event_type` The type of splicing event detected (e.g. `"se"`, `"ie"`, `"mxe"`, `"ri"`, `"si"`, `"a5ss"`, `"a3ss"`, `"atss"`, `"ates"`).

`event_tx_id` Transcript ID of the paired transcript involved in the event.

`event_estimate` DTU coefficient of the paired transcript.

`event_<col>` One column per name in `metadata(gr)$additional_columns`, prefixed with `event_`, carrying the corresponding value from the paired transcript.

`find_se()`: skipped exons

`find_ie()`: included exons

`find_mxe()`: mutually exclusive exons

`find_ri()`: retained introns

`find_si()`: spliced introns

`find_a5ss()`: alternative 5' splice sites

`find_a3ss()`: alternative 3' splice sites

`find_atss()`: alternative transcription start sites — first exons that begin transcription at a different coordinate

`find_ates()`: alternative transcription end sites — last exons that end transcription at a different coordinate

`find_all_events()`: all detected events

Examples

```
# make some mock data and run the function
gr <- create_mock_data(n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4) |>
  preprocess(coef_col = "estimate") |>
  generate_se(n_events = 1)

# this should find the skipped exon events we generated
find_se(gr, type = "boundary")

find_ie(gr, type = "boundary")


# detect mutually exclusive exons
gr_mx <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
) |>
  preprocess(coef_col = "estimate") |>
  generate_mxe(n_events = 1)

find_mxe(gr_mx, type = "boundary")


# detect retained introns
gr_ri <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
) |>
  preprocess(coef_col = "estimate") |>
  generate_ri(n_events = 1)

find_ri(gr_ri)

# the same data with the signs flipped: the introns retained above are
# now spliced out by the up transcripts
gr_si <- gr_ri |> dplyr::mutate(estimate = -estimate)
find_si(gr_si)


# detect alternative 5' splice sites
gr_a5 <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
) |>
  preprocess(coef_col = "estimate") |>
```

```
  generate_a5ss(n_events = 1)

find_a5ss(gr_a5)


# detect alternative 3' splice sites
gr_a3 <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
) |>
  preprocess(coef_col = "estimate") |>
  generate_a3ss(n_events = 1)
find_a3ss(gr_a3)


gr_tss <- data.frame(
  seqnames = "chr1",
  start = c(1, 21, 41, 5, 21, 41),
  end = c(10, 30, 50, 10, 30, 50),
  strand = "+",
  gene_id = "g1",
  tx_id = rep(c("down", "up"), each = 3),
  exon_rank = rep(seq_len(3), 2),
  estimate = rep(c(-1, 1), each = 3)
) |>
  plyranges::as_granges() |>
  preprocess(coef_col = "estimate")

find_atss(gr_tss)


# detect all event types at once
gr_all <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
) |>
  preprocess(coef_col = "estimate") |>
  generate_se(n_events = 1)

find_all_events(gr_all, type = "boundary", verbose = FALSE)
```

`generate_events` *Generate mock splice events in a GRanges object*

Description

Functions to introduce specific types of alternative splicing events into mock GRanges data for testing purposes.

Usage

```
generate_se(gr, n_events = 1)

generate_mxe(gr, n_events = 1)

generate_ri(gr, n_events = 1)

generate_a5ss(gr, n_events = 1)

generate_a3ss(gr, n_events = 1)

generate_atss(gr, n_events = 1, mode = c("shift", "drop"))

generate_ates(gr, n_events = 1, mode = c("shift", "drop"))
```

Arguments

gr	A GRanges object with metadata columns: 'exon_rank', 'gene_id', 'tx_id', and 'estimate'.
n_events	Number of events to generate
mode	How to move the site: `"shift"` moves the terminal exon's outer boundary (so it still overlaps its partner), `"drop"` removes the terminal exon and re-ranks (so the new terminal exon does not overlap its partner at all).

Value

`generate_se()`: A GRanges object with skipped exon events introduced

`generate_mxe()`: A GRanges object with mutually exclusive exon events introduced

`generate_ri()`: A GRanges object with retained intron events introduced

`generate_a5ss()`: A GRanges object with alternative 5' splice site events introduced

`generate_a3ss()`: A GRanges object with alternative 3' splice site events introduced

`generate_atss()`: A GRanges object with alternative transcription start site events introduced

`generate_ates()`: A GRanges object with alternative transcription end site events introduced

Examples

```
gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_se(gr, n_events = 1)


gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_mxe(gr, n_events = 1)
```

```
gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_ri(gr, n_events = 1)


gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_a5ss(gr, n_events = 1)


gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_a3ss(gr, n_events = 1)


gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_atss(gr, n_events = 1)

# the non-overlapping variant: drop the first exon altogether
generate_atss(gr, n_events = 1, mode = "drop")


gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4
)
generate_ates(gr, n_events = 1)
```

`get_seq` *Extract sequences for GRanges*

Description

Given a GRanges, look up the DNA or RNA sequence of each range in a reference genome. This is a simple helper function that rearranges the `getSeq` arguments and combines it with other helper functions to refer to genomes by name.

Usage

```
get_seq(gr, genome, as_rna = FALSE)
```

Arguments

gr	A GRanges object.
genome	Either a character string naming an installed BSgenome package (e.g. "hg38", or "BSgenome.Hsapiens.UCSC.hg38") or a BSgenome object. See ?getBSgenome for details.
as_rna	If TRUE, return an RNAStringSet (T -> U). Default FALSE returns a DNAStringSet.

Value

A DNAStringSet (or RNAStringSet if as_rna = TRUE), one entry per range in gr, in the same order. Assign it onto gr (e.g. gr$seq <- ...) to keep it as a metadata column.

Examples

```
gr <- GenomicRanges::GRanges(
  "chr1", IRanges::IRanges(start = c(1e6, 1.1e6), width = 50)
)

suppressPackageStartupMessages(
  library(Biostrings)
)

get_seq(gr, "hg38")
get_seq(gr, "hg38", as_rna = TRUE)

set.seed(123)
gr <- create_mock_data(
  n_genes = 2, n_tx_per_gene = 3, n_exons_per_tx = 6
) |>
  generate_se(n_events = 1) |>
  GenomicRanges::shift(50e6) # move mock data

skipped <- find_se(gr)

suppressPackageStartupMessages(
  library(magrittr)
)
# magrittr pipe needed for the `.` placeholder below
skipped %>%
  plyranges::flank_upstream(100) %>%
  dplyr::mutate(seq = get_seq(., "hg38"))
```

---

plot_transcript_lfc *Plot a per-transcript fold change for the transcripts of splicelogic events*

Description

Draws one diverging bar per transcript appearing in `tx_id` or `event_tx_id` of `events`: left of zero for a negative value and right for a positive one. It is the companion to `plot_transcripts()`: given the same `events` the two plots share their row order and y limits, so they line up when placed side by side, e.g. with `patchwork`.

Usage

```
plot_transcript_lfc(
  events,
  tx_lfc,
  title = NULL,
  lfc_max = NULL,
  lfc_label = "log2 fold change",
  lfc_ends = NULL,
  bar_color = "grey30",
  transcript_label = FALSE
)
```

Arguments

`events`	A GRanges returned by `find_all_events()` or any `find_*()` function.
`tx_lfc`	Numeric vector named by transcript id, e.g. a log2 fold change from a differential expression analysis. Transcripts not named in it keep their row but get no bar.
`title`	Plot title. Default `NULL`.
`lfc_max`	Half-width of the x axis. `NULL` (default) uses the largest absolute value in the plot; pass a number to make plots comparable. Values beyond it are clipped at the axis edge.
`lfc_label`	x-axis title.
`lfc_ends`	Optional length-2 character vector naming the negative and positive ends of the axis, e.g. `c("KD", "WT")` for a `log2(WT/KD)`. Default `NULL` draws no end labels.
`bar_color`	Colour of the bars. Default `"grey30"`.
`transcript_label`	Whether to print the transcript ids on the y axis, for use on its own. Default `FALSE`, as `plot_transcripts()` already labels the rows.

Value

A `ggplot` object.

Examples

```
exons <- data.frame(
  seqnames = "chr1",
  start = c(1, 201, 401, 1, 401),
  end = c(100, 300, 500, 100, 500),
```

```
  strand = "+",
  gene_id = "g1",
  tx_id = c("down", "down", "down", "up", "up"),
  exon_rank = c(1, 2, 3, 1, 2),
  estimate = c(-1, -1, -1, 1, 1)
) |>
  plyranges::as_granges() |>
  preprocess(coef_col = "estimate")

events <- find_all_events(exons, verbose = FALSE)
tx_lfc <- c(down = -1.8, up = 0.6)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_transcript_lfc(events, tx_lfc, transcript_label = TRUE)
}
```

`plot_transcripts` *Plot the transcripts involved in splicelogic events*

Description

Draws one row per transcript appearing in `tx_id` or `event_tx_id` of `events`. Transcripts are coloured by the sign of their estimate (blue when positive, red when negative), and the exons called as events are filled by event type, outlined, and labelled with the type. An exon called as several types at once, e.g. an a5ss and an a3ss when both of its boundaries moved, is filled with the first type and striped diagonally with the others. A bent arrow over the first exon marks where each transcript starts and which way it runs.

Usage

```
plot_transcripts(
  events,
  exons,
  title = NULL,
  up_color = "#1f77b4",
  down_color = "#d62728",
  event_colors = event_palette(),
  unknown_color = "#999999",
  rescale_introns = TRUE,
  new_intron_length = 50,
  flanking_length = c(50, 50),
  transcript_label = TRUE,
  label_events = TRUE,
  label_size = 2.6,
  legend = TRUE,
  tx_annot = NULL,
  stripe_multi_events = TRUE
)
```

Arguments

events	A GRanges returned by find_all_events() or any find_*() function.
exons	A flat GRanges of exons with a tx_id column holding every transcript in events, e.g. the preprocessed GRanges the events were found in.
title	Plot title. Default NULL.
up_color, down_color	Colours for transcripts with a positive and a negative estimate.
event_colors	Named vector mapping event type to fill colour. Default event_palette().
unknown_color	Fill for event types absent from event_colors.
rescale_introns	Whether gaps between exons are shortened to new_intron_length. Default TRUE.
new_intron_length	Length (bp) of each gap after rescaling. Default 50.
flanking_length	Length (bp) of the region drawn before the first and after the last exon. Default c(50, 50).
transcript_label	Whether to print the transcript id and estimate above each row. Default TRUE.
label_events	Whether to print the event type below each event exon. Default TRUE.
label_size	Size of the event labels. Default 2.6.
legend	Whether to draw the fill legend. Default TRUE.
tx_annot	Optional character vector named by transcript id, appended to that transcript's label (e.g. novel status). Transcripts not named in it keep the plain label.
stripe_multi_events	Whether an exon called as several event types is striped with the types its fill leaves out. Default TRUE; FALSE fills it with the first type alone, and drops the others from the legend.

Details

All events passed in are drawn on one panel, so they must come from a single gene; events from several genes are an error. Subset first, or use plot_transcripts_by_gene() for one plot per gene and plot_transcripts_by_pair() for one plot per transcript pair. A single event, e.g. plot_transcripts(events[1], exons), draws its two transcripts.

A per-transcript value such as a fold change is drawn separately by plot_transcript_lfc(). Given the same events, the two plots share their row order and y limits, so they line up side by side.

Value

A ggplot object.

Examples

```
# one gene: "down" has three exons, "up" skips the middle one
exons <- data.frame(
  seqnames = "chr1",
  start = c(1, 201, 401, 1, 401),
  end = c(100, 300, 500, 100, 500),
  strand = "+",
  gene_id = "g1",
  tx_id = c("down", "down", "down", "up", "up"),
  exon_rank = c(1, 2, 3, 1, 2),
  estimate = c(-1, -1, -1, 1, 1)
) |>
  plyranges::as_granges() |>
  preprocess(coef_col = "estimate")

events <- find_all_events(exons, verbose = FALSE)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_transcripts(events, exons)
}
```

---

plot_transcripts_by_gene

Plot the transcripts involved in splicelogic events, one plot per gene

---

Description

Splits `events` by `gene_id` and draws each gene with `plot_transcripts()`.

Usage

```
plot_transcripts_by_gene(events, exons, genes = NULL, title = NULL, ...)
```

Arguments

`events`	A GRanges returned by `find_all_events()` or any `find_*()` function, with a `gene_id` column.
`exons`	A flat GRanges of exons with a `tx_id` column holding every transcript in `events`, e.g. the preprocessed GRanges the events were found in.
`genes`	Optional character vector of gene ids to restrict to.
`title`	Optional prefix for the plot titles, which are otherwise the gene id.
`...`	Passed to `plot_transcripts()`.

Value

A named list of `ggplot` objects, one per gene.

Examples

```
exons <- data.frame(
  seqnames = "chr1",
  start = c(1, 201, 401, 1, 401),
  end = c(100, 300, 500, 100, 500),
  strand = "+",
  gene_id = "g1",
  tx_id = c("down", "down", "down", "up", "up"),
  exon_rank = c(1, 2, 3, 1, 2),
  estimate = c(-1, -1, -1, 1, 1)
) |>
  plyranges::as_granges() |>
  preprocess(coef_col = "estimate")

events <- find_all_events(exons, verbose = FALSE)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plots <- plot_transcripts_by_gene(events, exons)
  plots[["g1"]]
}
```

---

plot_transcripts_by_pair

Plot the transcripts involved in splicelogic events, one plot per pair

---

Description

Splits `events` by transcript pair, so each plot shows two transcripts and every event between them. The pair is unordered, so events reported from either side land in the same plot.

Usage

```
plot_transcripts_by_pair(events, exons, title = NULL, ...)
```

Arguments

`events`	A GRanges returned by `find_all_events()` or any `find_*()` function. A `tx_pair` column, if present, is used as the grouping as is.
`exons`	A flat GRanges of exons with a `tx_id` column holding every transcript in `events`, e.g. the preprocessed GRanges the events were found in.
`title`	Optional prefix for the plot titles, which are otherwise the transcript pair.
`...`	Passed to `plot_transcripts()`.

Value

A named list of `ggplot` objects, one per transcript pair.

Examples

```
exons <- data.frame(
  seqnames = "chr1",
  start = c(1, 201, 401, 1, 401),
  end = c(100, 300, 500, 100, 500),
  strand = "+",
  gene_id = "g1",
  tx_id = c("down", "down", "down", "up", "up"),
  exon_rank = c(1, 2, 3, 1, 2),
  estimate = c(-1, -1, -1, 1, 1)
) |>
  plyranges::as_granges() |>
  preprocess(coef_col = "estimate")

events <- find_all_events(exons, verbose = FALSE)

if (requireNamespace("ggplot2", quietly = TRUE)) {
  plots <- plot_transcripts_by_pair(events, exons)
  plots[["down-up"]]
}
```

---

prepare_exons *Prepare exon ranges from a TxDb and DTU results table*

Description

Extracts exon ranges from a TxDb object, merges them with differential transcript usage (DTU) results, and returns a flat GRanges ready for `preprocess`.

Usage

```
prepare_exons(
  txdb,
  dtu_table,
  coef_col,
  tx_id_col = "tx_id",
  gene_id_col = "gene_id",
  verbose = TRUE
)
```

Arguments

`txdb`	A `TxDb` object (from GenomicFeatures).
`dtu_table`	A data.frame or tibble with DTU results. Must contain columns for transcript ID, gene ID, and a coefficient.
`coef_col`	Column name in `dtu_table` with the coefficient / effect size values.

tx_id_col	Column name in dtu_table with transcript IDs matching the TxDb transcript names. Default "tx_id".
gene_id_col	Column name in dtu_table with gene IDs. Default "gene_id".
verbose	Whether to print progress messages. Default TRUE.

Value

A GRanges object with metadata columns: gene_id, tx_id, exon_rank, the coefficient column, and any additional columns from dtu_table.

Examples

```
library(AnnotationHub)
library(AnnotationDbi)
library(GenomicFeatures)
library(tibble)

ah <- AnnotationHub()
txdb <- ah[["AH84134"]] # fly TxDb (Drosophila melanogaster)

# build a simulated DTU table from the TxDb transcripts
txps <- txdb |>
  AnnotationDbi::select(
    keys(txdb, "TXID"), c("TXNAME", "GENEID"), "TXID"
  ) |>
  tibble::as_tibble() |>
  dplyr::select(tx_id = TXNAME, gene_id = GENEID)|>
  dplyr::filter(!is.na(gene_id))

sim_dtu_table <- txps |>
  dplyr::mutate(
    padj = runif(dplyr::n()),
    effect_est = rnorm(dplyr::n())
  )

fly_exons <- prepare_exons(
  txdb, sim_dtu_table, coef_col = "effect_est", verbose = TRUE
)
```

---

prepare_exons_by_partition

Prepare exons from two transcript partitions

Description

Combines two transcript partitions (up- and down-regulated) and assigns an estimate coefficient: +1 to up and -1 to down. Accepts either GRanges objects or character vectors of transcript IDs (in which case txdb is required to look up exon coordinates). The result is ready to pass to preprocess with coef_col = "estimate".

Usage

```
prepare_exons_by_partition(
  up,
  down,
  txdb = NULL,
  tx_id_col = "TXNAME",
  verbose = TRUE
)
```

Arguments

up	A GRanges object or character vector of transcript IDs for the upregulated partition (assigned `estimate = +1`).
down	A GRanges object or character vector of transcript IDs for the downregulated partition (assigned `estimate = -1`).
txdb	A `TxDb` object (from GenomicFeatures). Required when up and down are character vectors of transcript IDs.
tx_id_col	The keytype in `txdb` matching the transcript IDs in up and down. Default `"TXNAME"`. Only used when up and down are character vectors. See `AnnotationDbi::keytypes(txdb)` for available options.
verbose	Whether to print progress messages. Default `TRUE`.

Details

When up and down are GRanges, both must have `exon_rank`, `gene_id`, and `tx_id` metadata columns. Extra columns are kept; if one object lacks a column present in the other, those entries receive `NA`.

Value

A combined GRanges object with an `estimate` column (+1 for up, -1 for down), ready for `preprocess`.

Examples

```
# GRanges input
gr <- create_mock_data(n_genes = 1, n_tx_per_gene = 4, n_exons_per_tx = 4)
gr <- generate_se(gr, n_events = 1)
gr_down <- gr[gr$estimate < 0]
gr_up <- gr[gr$estimate > 0]
prepare_exons_by_partition(gr_up, gr_down) |>
  preprocess(coef_col = "estimate")
```

`preprocess` *Preprocess input GRanges object for splicing event calculation*

Description

This function checks that the input is a valid GRanges object with required metadata columns, then adds a unique key, the number of exons per transcript, and an 'internal' flag for each exon.

Usage

```
preprocess(gr, coef_col, method_string = NULL, additional_columns = NULL)
```

Arguments

`gr`	A GRanges object with metadata columns: 'exon_rank', 'gene_id', 'tx_id', 'coef'.
`coef_col`	The name of the metadata column indicating upregulated (+1) and downregulated (-1) exons.
`method_string`	The Differential Transcript Usage (DTU) method used to obtain the coef_col, for annotation purposes (optional).
`additional_columns`	A character vector of metadata column names to record for downstream use. Stored in `metadata(result)$additional_columns` (optional).

Value

A GRanges object with added 'key', 'nexons', and 'internal' columns.

Examples

```
# create mock data and run preprocessing
gr <- create_mock_data(n_genes = 2, n_tx_per_gene = 4, n_exons_per_tx = 4) |>
  preprocess(coef_col = "estimate", method_string = "mock_method")
```

Index

create_mock_data, 3

event_palette, 4
event_palette(), *14*

find_a3ss *(*find_events*)*, 4
find_a5ss *(*find_events*)*, 4
find_all_events *(*find_events*)*, 4
find_all_events(), *12*, *14–16*
find_alternative_3_prime_splice_sites *(*find_events*)*, 4
find_alternative_5_prime_splice_sites *(*find_events*)*, 4
find_alternative_transcription_end_sites *(*find_events*)*, 4
find_alternative_transcription_start_sites *(*find_events*)*, 4
find_ates *(*find_events*)*, 4
find_atss *(*find_events*)*, 4
find_events, 4
find_ie *(*find_events*)*, 4
find_included_exons *(*find_events*)*, 4
find_mutually_exclusive_exons *(*find_events*)*, 4
find_mxe *(*find_events*)*, 4
find_retained_introns *(*find_events*)*, 4
find_ri *(*find_events*)*, 4
find_se *(*find_events*)*, 4
find_si *(*find_events*)*, 4
find_skipped_exons *(*find_events*)*, 4
find_spliced_introns *(*find_events*)*, 4

generate_a3ss *(*generate_events*)*, 8
generate_a5ss *(*generate_events*)*, 8
generate_ates *(*generate_events*)*, 8
generate_atss *(*generate_events*)*, 8
generate_events, 8
generate_mxe *(*generate_events*)*, 8
generate_ri *(*generate_events*)*, 8
generate_se *(*generate_events*)*, 8
GenomicRanges::findOverlaps(), *6*
get_seq, 10

plot_transcript_lfc, 11
plot_transcript_lfc(), *14*
plot_transcripts, 13
plot_transcripts(), *4*, *12*, *15*, *16*
plot_transcripts_by_gene, 15
plot_transcripts_by_gene(), *14*
plot_transcripts_by_pair, 16
plot_transcripts_by_pair(), *14*
prepare_exons, 17
prepare_exons_by_partition, 18
preprocess, *17–19*, 20

splicelogic *(*splicelogic-package*)*, 2
splicelogic-package, 2