

Package ‘splicelogic’

June 22, 2026

Title splicelogic: differential transcripts to splice events

Version 1.0.1

Description Translate differential transcript usage results into discrete splice events.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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)

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

Suggests knitr, rmarkdown, testthat (>= 3.0.0), readr, wiggleplotr,
GenomicFeatures, AnnotationHub, ggplot2, AnnotationDbi, Seqinfo

Config/testthat/edition 3

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/splicelogic>

git_branch RELEASE_3_23

git_last_commit eecaf3c

git_last_commit_date 2026-05-13

Repository Bioconductor 3.23

Date/Publication 2026-06-21

Author Beatriz Campillo [aut, cre] (ORCID:

<https://orcid.org/0000-0001-7323-9125>),

Michael Love [aut] (ORCID: <https://orcid.org/0000-0001-8401-0545>),

NIH NHGRI [fnd],

Wellcome Trust [fnd]

Maintainer Beatriz Campillo <beatrizcampillo29@gmail.com>

Contents

splicellogic-package	2
create_mock_data	3
find_events	3
generate_events	6
prepare_exons	7
prepare_exons_by_partition	9
preprocess	10

Index **11**

splicellogic-package *splicellogic: differential transcripts to splice events*

Description

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

Author(s)

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

Authors:

- Michael Love <michaelisaiahlove@gmail.com> ([ORCID](#))

Other contributors:

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

See Also

Useful links:

- <https://github.com/thelovelab/splicellogic>
- <https://thelovelab.github.io/splicellogic>
- Report bugs at <https://github.com/thelovelab/splicellogic/issues>

create_mock_data	<i>Create mock GRanges data for splicing event testing</i>
------------------	--

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)  
)
```

Arguments

n_genes	Number of genes to simulate
n_tx_per_gene	Number of transcripts per gene
n_exons_per_tx	Number of exons per transcript
coef_range	Range of coefficient values to sample from

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)
```

find_events	<i>Find splice events from annotated exons</i>
-------------	--

Description

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

Usage

```

find_se(gr, type = c("boundary", "over", "in"), inverse = FALSE)

find_ie(gr, type = c("boundary", "over", "in"))

find_mxe(gr, type = c("boundary", "in", "over"))

find_ri(gr)

find_a5ss(gr)

find_a3ss(gr)

find_all_events(gr, type = c("boundary", "over", "in"), verbose = TRUE)

```

Arguments

<code>gr</code>	A GRanges object with exon annotations, including 'tx_id', 'exon', and 'coef_col' metadata columns and preprocessed with preprocess().
<code>type</code>	The type of overlap to consider when identifying events.
<code>inverse</code>	If TRUE, identifies included exons instead of skipped exons.
<code>verbose</code>	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", "a5ss", "a3ss").

`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_a5ss()`: alternative 5' splice sites

`find_a3ss()`: alternative 3' splice sites

`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)

# 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)

# 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	<i>Generate mock splice events in a GRanges object</i>
-----------------	--

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)
```

Arguments

gr	A GRanges object with metadata columns: 'exon_rank', 'gene_id', 'tx_id', and 'estimate'.
n_events	Number of events to generate

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

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)
```

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 <code>AnnotationDbi::keytypes(txdb)</code> 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

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

`find_a3ss` (`find_events`), [3](#)
`find_a5ss` (`find_events`), [3](#)
`find_all_events` (`find_events`), [3](#)
`find_events`, [3](#)
`find_ie` (`find_events`), [3](#)
`find_mxe` (`find_events`), [3](#)
`find_ri` (`find_events`), [3](#)
`find_se` (`find_events`), [3](#)

`generate_a3ss` (`generate_events`), [6](#)
`generate_a5ss` (`generate_events`), [6](#)
`generate_events`, [6](#)
`generate_mxe` (`generate_events`), [6](#)
`generate_ri` (`generate_events`), [6](#)
`generate_se` (`generate_events`), [6](#)

`prepare_exons`, [7](#)
`prepare_exons_by_partition`, [9](#)
`preprocess`, [7](#), [9](#), [10](#)

`splicelogic` (`splicelogic-package`), [2](#)
`splicelogic-package`, [2](#)