Package ‘BREW3R.r’

May 3, 2024

Type Package
Title R package associated to BREW3R
Version 1.0.0
Description This R package provide functions that are used in the BREW3R workflow. This mainly contains a function that extend a gtf as GRanges using information from another gtf (also as GRanges). The process allows to extend gene annotation without increasing the overlap between gene ids.
License GPL-3
Encoding UTF-8
LazyData false
RoxygenNote 7.3.1
Imports GenomicRanges, methods, rlang, S4Vectors, utils
Suggests testthat (>= 3.0.0), IRanges, knitr, rmarkdown, BiocStyle, rtracklayer
biocViews GenomeAnnotation
Config/testthat/edition 3
URL https://github.com/lldelisle/BREW3R.r
BugReports https://github.com/lldelisle/BREW3R.r/issues/
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/BREW3R.r
git_branch RELEASE_3_19
git_last_commit 04ece77
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-03
Author Lucille Lopez-Delisle [aut, cre]
Maintainer Lucille Lopez-Delisle <lucille.delisle@epfl.ch>
add_new_exons

Description

A function that from 2 GRanges add exons from the second one to the first one if the 3p of the last exon of the transcript in the first GRanges matches the 3p of an exon in the second one.

Usage

add_new_exons(input_gr_to_extend, input_gr_with_new_exons)

Arguments

input_gr_to_extend
A GRanges to be complemented

input_gr_with_new_exons
A GRanges with exons to be added to the first one (exons with strand '*' are excluded)

Details

Potential new exons will be filtered for collision with exons present in the first GRanges even if they belong to the same gene_id. For the moment all potential exons extensions are added to the same existing transcript_id so introns maybe artificial introns.

Value

A GRanges identical to 'input_gr_to_extend' with new exons whose 'exon_id' contains BREW3R. 'exon_number' may have changed.
**adjust_for_collision**

*Description*

A function that from a GRanges with 'old_width' Change the starts and ends to prevent collisions larger than with old coordinates

*Usage*

```r
adjust_for_collision(input_gr)
```

*Arguments*

- **input_gr**: A GRanges with 1 meta: 'old_width'

*Value*

A list with:
- 'pot_issues': A dataframe with exons which overlaps between 'input_gr' and itself while gene_ids are different
- 'new_gr': A GRanges identical to 'input_gr' except that start/end have been adjusted to prevent collisions.

**debug_msg**

*Display debug messages if verbose allows it*

*Description*

A function that extend rlang::inform to display a message if the verbose is at "debug" and show content of the variable

*Usage*

```r
debug_msg(message = NULL, ...)
```

*Arguments*

- **message**: String to display
- **...**: Other parameters for rlang::inform

*Value*

Nothing
extend_granges

**Description**
A function that from a GRanges from gtf will extend the 3’ of transcripts using another GRanges from gtf as a template

**Usage**
```r
extend_granges(
  input_gr_to_extend,
  input_gr_to_overlap,
  extend_existing_exons = TRUE,
  add_new_exons = TRUE,
  overlap_resolution_fn = NULL
)
```

**Arguments**
- `input_gr_to_extend`: A GRanges to extend (only exons are kept and strand * are excluded)
- `input_gr_to_overlap`: A GRanges with intervals to overlap
- `extend_existing_exons`: A boolean that indicates if existing exons should be extended
- `add_new_exons`: A boolean that indicates if new exons with compatible splicing event should be added
- `overlap_resolution_fn`: A file path where the dataframe giving details on the collision resolution is written

**Details**
During the extension process a special care is taking to prevent extension which would lead to overlap between different gene_ids.

**Value**
A GRanges based on ‘input_gr_to_extend’ where exons are extended and new exons can be added. Exons extended will have a ’.ext’ suffix to the original exon_id. Exons added will have a exon_id starting with ‘BREW3R’.
### Examples

```
# Very simple case
# input_gr:         -------> ----->
# to_overlap:      ------------------>
# output:         ----------------> ----->

input_gr <- GenomicRanges::GRanges(  
  seqnames = "chr1",  
  ranges = IRanges::IRanges(start = c(5, 20),  
    end = c(10, 30)),  
  strand = "+",  
  gene_id = c("gene1", "gene2"),  
  transcript_id = c("transcript1", "transcript2"),  
  type = "exon",  
  exon_id = c("exon1", "exon2")
)

input_gr_to_overlap <- GenomicRanges::GRanges(  
  seqnames = "chr1",  
  ranges = IRanges::IRanges(start = 3,  
    end = 15),  
  strand = "+",  
  gene_id = "geneA",  
  transcript_id = "transcriptA",  
  type = "exon",  
  exon_id = "exonA"
)

extend_granges(input_gr, input_gr_to_overlap)
```

### Description

A function that from 2 GRanges returns a subset of the first GRanges which have been extended using the second GRanges.

### Usage

```
extend_using_overlap(input_gr_to_extend, input_gr_to_overlap)
```

### Arguments

- **input_gr_to_extend**
  
  A GRanges with exons to extend (strand * are excluded)

- **input_gr_to_overlap**
  
  A GRanges with intervals to overlap
**filter_new_exons**

**Value**

A GRanges which is a subset of ‘input_gr_to_extend’ where 3’ end have been modified to match the 3’ end of ‘input_gr_to_overlap’ if they overlap (initial width have been stored into old_width)

---

**extract_last_exons**

**Extract last exons**

**Description**

A function that from a GRanges from gtf select only entries for the last exons. If multiple exons overlap the last base of the grouping_variable, they will all be reported.

**Usage**

```r
extract_last_exons(
  input_gr,
  grouping_variable = "transcript_id",
  invert = FALSE
)
```

**Arguments**

- `input_gr` A GRanges from a gtf
- `grouping_variable` A string with the name of the metadata which should be used to group
- `invert` A boolean that indicates if you want all except the last exons

**Value**

A GRanges which contains a subset of ‘input_gr’

---

**filter_new_exons**

**Filter new exons for collision**

**Description**

A function that from 2 GRanges filter exons from the first one so they do not go three prime to the first collision with the second one.

**Usage**

```r
filter_new_exons(all_exons_interesting, input_gr_to_extend)
```
five_prime_pos

Arguments

  all_exons_interesting
    A GRanges with exons to trim and filter
  input_gr_to_extend
    A GRanges to overlap

Value

  A GRanges subset of 'all_exons_interesting'

Description

  A function that from a GRanges gives the 5’ position

Usage

  five_prime_pos(input_gr)

Arguments

  input_gr
    A GRanges or GRangesList

Value

  A vector of integers

overlap_different_genes

  Get overlaps from different genes

Description

  A function that from 2 GRanges generates a dataframe With queryHits, subjectHits when the
gene_id is different

Usage

  overlap_different_genes(gr1, gr2)

Arguments

  gr1
    A GRanges with 'gene_id'
  gr2
    A GRanges with 'gene_id'
Value

A data frame with overlaps between gr1 and gr2 when gene_id from gr1 is different from gene_id from gr2. The data frame has 4 columns: ‘queryHits’, ‘subjectHits’, ‘query_gene_id’ and ‘subject_gene_id’.

progression_msg

Display progression messages if verbose allows it

Description

A function that extend rlang::inform to display a message if the verbose is at "debug" or "progression”

Usage

progression_msg(...)

Arguments

... Parameters for rlang::inform

Value

Nothing

three_prime_pos

Get three prime position

Description

A function that from a GRanges gives the 3’ position

Usage

three_prime_pos(input_gr)

Arguments

input_gr A GRanges or GRangeList

Value

A vector of integers
Index

add_new_exons, 2
adjust_for_collision, 3

debug_msg, 3

extend_granges, 4
extend_using_overlap, 5
extract_last_exons, 6

filter_new_exons, 6
five_prime_pos, 7

overlap_different_genes, 7
progression_msg, 8

three_prime_pos, 8