## Package ‘BREW3R.r’

May 29, 2024

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>R package associated to BREW3R</td>
</tr>
<tr>
<td>Version</td>
<td>1.0.1</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>LazyData</td>
<td>false</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>7.3.1</td>
</tr>
<tr>
<td>Imports</td>
<td>GenomicRanges, methods, rlang, S4Vectors, utils</td>
</tr>
<tr>
<td>Suggests</td>
<td>testthat (&gt;= 3.0.0), IRanges, knitr, rmarkdown, BiocStyle, rtracklayer</td>
</tr>
<tr>
<td>biocViews</td>
<td>GenomeAnnotation</td>
</tr>
<tr>
<td>Config/testthat/edition</td>
<td>3</td>
</tr>
<tr>
<td>URL</td>
<td><a href="https://github.com/lldelisle/BREW3R.r">https://github.com/lldelisle/BREW3R.r</a></td>
</tr>
<tr>
<td>BugReports</td>
<td><a href="https://github.com/lldelisle/BREW3R.r/issues/">https://github.com/lldelisle/BREW3R.r/issues/</a></td>
</tr>
<tr>
<td>VignetteBuilder</td>
<td>knitr</td>
</tr>
<tr>
<td>git_url</td>
<td><a href="https://git.bioconductor.org/packages/BREW3R.r">https://git.bioconductor.org/packages/BREW3R.r</a></td>
</tr>
<tr>
<td>git_branch</td>
<td>RELEASE_3_19</td>
</tr>
<tr>
<td>git_last_commit</td>
<td>55629ac</td>
</tr>
<tr>
<td>git_last_commit_date</td>
<td>2024-05-14</td>
</tr>
<tr>
<td>Repository</td>
<td>Bioconductor 3.19</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2024-05-29</td>
</tr>
<tr>
<td>Author</td>
<td>Lucille Lopez-Delisle [aut, cre] [<a href="https://orcid.org/0000-0002-1964-4960">https://orcid.org/0000-0002-1964-4960</a>]</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Lucille Lopez-Delisle <a href="mailto:lucille.delisle@epfl.ch">lucille.delisle@epfl.ch</a></td>
</tr>
</tbody>
</table>
Contents

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 ................................................ 7
five_prime_pos .................................................. 7
overlap_different_genes ....................................... 8
progression_msg ................................................ 8
three_prime_pos ................................................ 9

Index

add_new_exons  ................................................. 10

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  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
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.

display_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
display_msg(message = NULL, ...)

Arguments
message  String to display
...  Other parameters for rlang::inform

Value
Nothing
**extend_granges**

**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’.
extend_using_overlap

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)
```

extend_using_overlap  
Overlap exons and extend three prime end

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)
```
extract_last_exons

Arguments

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

input_gr_to_overlap
A GRanges with intervals to overlap

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)

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

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**

`filter_new_exons(all_exons_interesting, input_gr_to_extend)`

**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`

---

**five_prime_pos**

*Get five prime position*

**Description**

A function that from a GRanges gives the 5’ position

**Usage**

`five_prime_pos(input_gr)`

**Arguments**

- `input_gr`: A GRanges or GRangeList

**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**

```r
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**

```r
progression_msg(...)```

**Arguments**

- `...` Parameters for `rlang::inform`

**Value**

Nothing
three_prime_pos

---

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, 7
five_prime_pos, 7

overlap_different_genes, 8
progression_msg, 8

three_prime_pos, 9