Package ‘DuplexDiscovereR’

December 1, 2025

Title Analysis of the data from RNA duplex probing experiments

Description DuplexDiscovereR is a package designed for analyzing data from RNA cross-linking and proximity ligation protocols such as SPLASH, PARIS, LIGR-seq, and others.
DuplexDiscovereR accepts input in the form of chimerically or split-aligned reads. It includes procedures for alignment classification, filtering, and efficient clustering of individual chimeric reads into duplex groups (DGs). Once DGs are identified, the package predicts RNA duplex formation and their hybridization energies. Additional metrics, such as p-values for random ligation hypothesis or mean DG alignment scores, can be calculated to rank final set of RNA duplexes.
Data from multiple experiments or replicates can be processed separately and further compared to check the reproducibility of the experimental method.

License GPL-3

URL `https://github.com/Egors01/DuplexDiscovereR/`

BugReports `https://github.com/Egors01/DuplexDiscovereR/issues/`

Encoding UTF-8

NeedsCompilation no

Version 1.4.0

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

BiocType Software

biocViews Sequencing, Transcriptomics, StructuralPrediction, Clustering, SplicedAlignment

Imports Gviz, Biostrings, rtracklayer, GenomicAlignments, GenomicRanges, ggsci, igraph, rlang, scales, stringr, dplyr, tibble, tidyr, purrr, methods, grDevices, stats, utils, vctrs

Depends R (>= 4.5), InteractionSet

LazyData false

Suggests knitr, UpSetR, BiocStyle, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

VignetteEngine knitr

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/DuplexDiscovereR

git_branch RELEASE_3_22

git_last_commit 38c4e52

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-12-01

Author Egor Semenchenko [aut, cre, cph] (ORCID: <`https://orcid.org/0009-0007-5306-076X`>), Volodymyr Tsybulskyi [ctb] (ORCID: <`https://orcid.org/0009-0002-4141-6291`>), Irmtraud M. Meyer [aut, cph] (ORCID: <`https://orcid.org/0000-0002-4048-3479`>)

Maintainer Egor Semenchenko <`yegor.smb@gmail.com`>

Contents

`.addDGidsForTmpDGs` *Helper function to add ids to the duplex groups missed during global clustering*

Description

Check if there are a temporary duplex records with `duplex_id`, which consist of more than one read `n_reads` > 1 , but does not have assigned any `dg_id` as the duplex group (DG) index. Creates new `dg_id` if `n_reads` > 1

Usage

```
.addDGidsForTmpDGs(gi_input)
```

Arguments

`gi_input`	**GInteractions** with the `dg_id`, `duplex_id` and `n_reads` column

Details

Meant to be used in the situations when previous collapsing steps merged two or more reads to the temporary DG with `duplex_id`, but global clustering has not identified any overlap between this temporary group and other duplexes, resulting in undefined `dg_id`. This function looks up for these cases and creates new `dg_id` for temporary DGs, marking them as the final DGs. New `dg_id` values are unique and allocated sequentially after the maximum value of `dg_id`

Value

GInteractions object with new `dg_id` for rows with `n_reads` > 1

`.addGeneCounts` *Helper function to add count data to metadata of* `GInteractions`

Description

Merges the count dataframe and interactions metadata by `id_col` If key is not found, in metadata throws error

Usage

```
.addGeneCounts(gi, df_counts, id_col = "gene_id")
```

Arguments

`gi`	`GInteractions`
`df_counts`	dataframe with read counts
`id_col`	key to use in merge

Value

`GInteractions` with added counts

`.annotateCisTrans` *Annotate RNA-RNA interactions as cis- and trans-*

Description

Annotated each entry gi object as cis, if the .A and .B arms correspond to the same feature (i.e transcript_id or gene_id) If the values are are equal, then annotated with value: `cis` = 1, If not equal or `NA`: `cis` = 0

Usage

```
.annotateCisTrans(gi, id_col_base = "gene_id")
```

Arguments

`gi`	`GInteractions` object containing two metadata columns as feature annotation
`id_col_base`	base name of the feature id columns to use. Function will look for <id_col_base>.A and <id_col_base>.B columns and compare them

Value

gi `GInteractions` object containing `cis` field with 0/1 values

`.compute_clusters_comp` *Helper function to call clustering on each component*

Description

Helper function to call clustering on each component

Usage

```
.compute_clusters_comp(graph, index)
```

Details

Call clustering on each independent graph component. Used when decompose==TRUE

Value

sub-graph with clusters labelled with sample-wise unique ids (cluster_group)

`.DGIdToDuplexId` *Accessor for mapping between temporary and final cluster ids*

Description

Accessor for mapping between temporary and final cluster ids

Usage

```
.DGIdToDuplexId(gi)
```

Arguments

`gi`

Value

tibble

`.getStartEndOvl` *Helper func for calculating one-side overlap of SJ and junctions of gi*

Description

Helper func for calculating one-side overlap of SJ and junctions of gi

Usage

```
.getStartEndOvl(gi, gr_chim_c, gr_sj_c, tol = 3)
```

Arguments

`gi`	GInteractions object
`gr_chim_c`	GRanges object with chimeric junctions of gi
`gr_sj_c`	GRanges with chimeric junctions
`tol`	overlap tolerance

Value

Granges object with the left (A) region

`.gv_plotboxes` *Plots distributed boxes*

Description

Non-exported from Gviz, contains logic for calcualting boxes alignment on plot

Usage

```
.gv_plotboxes(box, lwd, lty, alpha)
```

Value

pushes boxes to the viewport

`.gv_updatepars` *Set Gviz graphical parameters*

Description

Set Gviz graphical parameters

Usage

```
.gv_updatepars(x, class)
```

Details

Non-exported from Gviz. Uses the same procedure as in AnnotationTrack to set defaults.

Value

set default values inside class upon calling withn class constructor

`annotateGI` *Annotate RNA duplexes with features*

Description

Overlays RNA duplexes with `GRanges` annotation object.

Usage

```
annotateGI(
  gi,
  anno_gr,
  keys = c("gene_name", "gene_type", "gene_id"),
  ambig_key = keys[1],
  save_ambig = TRUE,
  order_key = NULL,
  order_vec = NULL
)
```

Arguments

`gi`	`GInteraction` object to annotate
`anno_gr`	`GRanges` object with the `keys` columns in the metadata
`keys`	vectors with the names of the features to use for annotation.
`ambig_key`	a which feature to use for recording the annotation ambiguity. Determines the values in `ambig_list.A` and `ambig_list.B`.
`save_ambig`	When RNA duplex overlaps multiple features, ambiguous annotation for a single key can be stored in `ambig_list.A` and `ambig_list.B`. `ambig.A` and 'ambig.B' fields will be added as the 0/1 label

`order_key`	Experimental. In case RNA duplex overlaps multiple features, this key will be used to sort the overlapping features.
`order_vec`	Experimental. An ordered vector of values in `order_key` annotation feature, which sets the priority in case of feature overlap.

Details

For each annotation feature in `keys`, i.e if keys=c(keyname1), then `<keyname1>.A`, `<keyname1>.B` annotation fields will be created, containing the names of overlapping features If no overlap is found for the feature, then filed will have NA

Value

`GInteractions` object with new fields

Examples

```
data("RNADuplexesSampleData")
annotateGI(gi = RNADuplexSampleDGs, anno_gr = SampleGeneAnnoGR)

# Prioritisation of the snRNA and lncRNA before mRNA if genes overlap
annotateGI(
    gi = RNADuplexSampleDGs, anno_gr = SampleGeneAnnoGR,
    keys = c("gene_id", "gene_name", "gene_type"),
    order_key = "gene_type",
    order_vec = c("snRNA", "lncRNA", "protein_coding"),
    save_ambig = TRUE
)
```

`availableDisplayPars` *The default display parameters for a* `DuplexTrack` *object*

Description

`DuplexTrack` inherits from `[Gviz::Annotaiontrack()]` and its Gviz parents. Most likely, user doesn't need all dioplay pars for the parents, so only parameters relevant to the `DuplexTrack` are returned by default.

Usage

```
availableDisplayPars(class)
```

Arguments

`class`	`DuplexTrack` track object This function allows user to display the default display parameters for the `DuplexTrack` class.

Value

list of the default display parameters.

Examples

```
library(InteractionSet)
anchor1 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(100, 600, 1100, 1600, 2100),
        end = c(200, 700, 1200, 1700, 2200)
    ),
    strand = "+"
)
anchor2 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(300, 800, 1300, 1800, 2300),
        end = c(400, 900, 1400, 1900, 2400)
    ),
    strand = "+"
)

interactions <- GInteractions(anchor1, anchor2, mode = "strict")
gr_region <- range(anchor1, anchor2)
a <- DuplexTrack(interactions, gr_region = gr_region, stacking = "dense")
availableDisplayPars("DuplexTrack")
DuplexDiscovereR::availableDisplayPars(a)
```

---

calculateLigationPvalues

Calculate p-values and abundance fractions for RNA duplexes

Description

Calculates p-values by applying Fisher test to each gene/transcript pair Uses BH correction, outputs duplex abundance relative to the per - gene/transcript count, and counts of other RNA duplexes formed by either or none gene/transcript in this pair.

Usage

```
calculateLigationPvalues(gi, df_counts, id_col = "gene_id")
```

Arguments

`gi`	`GInteraction` object annotated with gene/transcript names
`df_counts`	`data.frame` A two- column dataframe with gene/transcript counts to. The first column should match the 'gene_id' feature in anno_gr. The second column is the respective count.
`id_col`	the prefix for gene/transcript metadata id fields in input gi. Two fields of <id_col>.A and <id.col>.B are expected. Otherwise throws error.

Details

H0: RNA duplex not existing and reported due to the random ligation of fragments H1: RNA duplex is true and formed because of existing the RNA-RNA interaction

The probability of random ligation is modeled as \(P(a, b)\) given by the following equation: The probability $P(a, b)$ is defined as:

$$P(a,b) \propto \begin{cases} 2 \cdot P(a) \cdot P(b) & \text{if } a : b \text{ is observed and } a \neq b \\ P(a) \cdot P(b) & \text{if } a : b \text{ is observed and } a = b \\ 0 & \text{else} \end{cases}$$

where The probability (P(a)) (same as for P(b)) is calculated as: $P(a) = \frac{\text{N reads(a)}}{\text{total N reads}}$

p-value calculated by comparing observed duplex abundance to the expected as the are under the curve distribution to the right of the observed. P(a, b) is normalized to sum up to one.

Value

`GInteractions` object with new fields

Examples

```
data("RNADuplexesSampleData")
gi <- calculateLigationPvalues(RNADuplexSampleDGs, df_counts = RNADuplexesGeneCounts)
hist(gi$p.adj, breaks = 20)
```

`classifyTwoArmChimeras` *Wrapper for classification of the 2arm chimeric reads*

Description

Wraps two procedures for different types of classification for read alignment:

overlap type test if chimeric junction map to two non-overlapped regions or shorter than defined minimum distance

splice junction test if chimeric junction is also a splice junction

Usage

```
classifyTwoArmChimeras(
  gi,
  min_junction_len = 4,
  junctions_gr,
  max_sj_shift = 4
)
```

Arguments

`gi`	`GInteractions` object
`min_junction_len`	minimum allowed distance between two chimeric arms
`junctions_gr`	`Granges` object with the splice junctions coordinates
`max_sj_shift`	maximum shift between either donor and acceptor splice sites and corresponding chimreic junction coordinates to count chimeric junction as splice junction

Details

Calls detection of the chimeric junction type, annotates short junctions on same chromosome an strand as 'short'. Compares chimeric junctions with splice junctions. Adds results as the new metadata fields parallel to the input.

Value

`GInteractions` object object of the same size with new columns:

splicejnc filled with 0 or 1

junction_type factor for the junction types

See Also

DuplexDiscovereR::getChimericJunctionTypes(), DuplexDiscovereR::getSpliceJunctionChimeras()

Examples

```
data("RNADuplexesSampleData")
head(RNADuplexSampleGI)
# remove all metadata
mcols(RNADuplexSampleGI) <- NULL
gi <- classifyTwoArmChimeras(RNADuplexSampleGI,
    min_junction_len = 5,
    junctions_gr = SampleSpliceJncGR, max_sj_shift = 10
)
table(gi$splicejnc)
table(gi$junction_type)
```

`clusterDuplexGroups` *Cluster RNA duplexes in* `GInteractions` *object*

Description

Main method to find duplex groups from the individual interactions

Usage

```
clusterDuplexGroups(
  gi,
  graphdf = NULL,
  maxgap = 40,
  minoverlap = 10,
  id_column = "duplex_id",
  weight_column = "weight",
  fast_greedy = FALSE,
  decompose = FALSE,
  id_columns_grapdf = paste(id_column, c(1, 2), sep = "."),
  min_arm_ratio = 0.3,
  dump_graph = FALSE,
  dump_path = ""
)
```

Arguments

`gi`	`GInteractions` object
`graphdf`	Optional. Dataframe representing connection edges between entries in gi If not provided, graphdf is created inside the function
`maxgap`	For graph creation only. Max shift between arms starts and ends for pair of overlapping reads
`minoverlap`	For graph creation only. Minimum required overlap between either arm for pair of overlapping reads Other optional arguments, which are not relevant, unless user want to modify clustering weights or modify clustering in some other way
`id_column`	Optional. Column name in the `GInteractions` metadata, which was used to index temporary duplex groups, if they are present
`weight_column`	Optional. If graphdf is provided, field to use for weight overlaps
`fast_greedy`	Optional. Run the fast_greedy algorithm instead of Louvain. Can speed up calcualtion for the large graphs.
`decompose`	Decompose graph into separate sub-graphs before clustering.
`id_columns_grapdf`	Column in the graph dataframe, which was used for index
`min_arm_ratio`	For graph creation only. Span-to-overlap ratio threshold. If smaller than this value, then edge is not drawn
`dump_graph`	For debug. Export the graph elements. not used
`dump_path`	For debug. PArt to export the graph elements. not used

Details

Accepts or creates the connections graphdf dataframe, creates graph with `igraph` package, uses community detection algoritm to call clusters. New field `dg_id` is added to label the clusters (duplex groups). If no community is found for the read, `dg_id` is `NA`

Value

`GInteractions` object with new `dg_id` column

Examples

```
data("RNADuplexesSampleData")
# run preprocessing and filtering
preproc_df <- runDuplexDiscoPreproc(RNADuplexesRawBed, table_type = "bedpe")
preproc_gi <- makeGiFromDf(preproc_df)
preproc_gi <- classifyTwoArmChimeras(preproc_gi,
    min_junction_len = 5,
    junctions_gr = SampleSpliceJncGR, max_sj_shift = 10
)
# collapse duplicates
gi <- collapseIdenticalReads(preproc_gi)$gi
# run global clustering
gi <- clusterDuplexGroups(gi)
# check dg_ids
table(is.na(gi$dg_id))
```

`collapseIdenticalReads`
Collapses identical interactions

Description

Two entries (reads) are considered identical if they share start, end, strand and score vales Identical entries are collapsed into the single one.

Usage

```
collapseIdenticalReads(gi)
```

Arguments

gi	GInteractions(mode='strict') object with chromA, strandA, startA, endA, chromB, strandB, startB, endB, score columns Optionally cigar_alnA, cigar_alnB columns are also considered for collapsing 'read_id' column used as the index in the initial objects. Created, if not exists

Details

Adds columns to the collapsed object duplex_id (int) unique record id n_reads (int) number of entries collapsed

Value

result_list object with keys ' gi_collapsed': New collapsed GInteraction object ' stats_df': tibble with the mapping of the original entries to the new duplex_id

Examples

```
# load data
data("RNADuplexesSmallGI")
res_collapse <- collapseIdenticalReads(SampleSmallGI)
gi_new <- res_collapse[["gi_collapsed"]]
# keeps the mapping of the colapsed object to new
read_stats_df <- res_collapse[["stats_df"]]
```

`collapseSimilarChimeras`
Call clustering multiple times to collapse similar reads into duplex groups

---

Description

Function calls clustering algorithm several times and collapses highly similar reads to the temporary duplex groups (DGs).

Usage

```
collapseSimilarChimeras(
  gi,
  read_stats_df,
  maxgap = 5,
  niter = 2,
  minoverlap = 10,
  min_nodes = 10
)
```

Arguments

`gi`	`GInteractions` object
`read_stats_df`	`tibble` with the mapping 'read_id' and 'duplex_id' fields 'read_id' refers to the unique read, 'duplex_id' refers to the entry collapsed identical reads i.e two identical reads will will correspond to two unique read_id and the single duplex_id with n_reads=2
`maxgap`	Maximum relative shift between the overlapping read arms
`niter`	Number of times clustering will be called
`minoverlap`	Minimum required overlap between either read arm
`min_nodes`	Minimum count of nodes to finish the interaction merging

Details

Calling this procedure before global read clustering substantially reduces time required for calling DGs. Collapsed duplex groups are aggregated only from the reads which are shifted by only a few nucleotides from each other. These DGs are temporary until full library clustering is called. To keep track of the mapping of the temprary DGs to the input, dedicated dataframe is returned. The 'duplex_id' column will be added or updated as identifier for the temporary duplex group. The number of reads under single 'duplex_id' is recorded in the 'n_reads' fields

Value

a list with the following keys

gi_updated `GInteractions` object with both collapsed duplex groups and not-collapsed unchanged reads

stats_df `tibble` With the mapping from the unique read - with the the infromation about time and memory reaquired for the function call

---

`collapse_duplex_groups` *Collapse the reads into the duplex groups after clustering*

Description

Collapse each interaction in the input to the duplex group based on the pre-computed dg_id

Usage

```
collapse_duplex_groups(
  gi,
  return_unclustered = FALSE,
  return_collapsed = TRUE,
  keep_meta = TRUE
)
```

Arguments

`gi`	`GInteractions` with the 'dg_id' metadata field
`return_unclustered`	add unclustered reads to output
`return_collapsed`	add duplex groups, which were created as temporary with n_reads > 1 but was not clustered to the DG golabally. This parameter is used internally and should be kept default in most situations.
`keep_meta`	whether to keep metadata, which only unclustered reads have, in case of a mixed output

Details

'dg_id' is used as the identifier for the duplex group Reads belonging to the same duplex group are collapsed into a single entry with start and end are set as min() and max() coordinate of the reads in within the duplex group. The 'score' column is averaged across the duplex group reads is calculated and put as the 'score' for the collapsed duplex group Behavior in case 'dg_id' = NA: Option '`return_unclustered`' - whether unclustered reads with should be added to the output gi

return_unclustered == FALSE Interaction is not returned in the output. Default.

return_unclustered == TRUE Interaction is returned in the output, output is mixed duplex groups and individual reads

Internally used argument #'

return_collapsed == FALSE In case interaction already collapsed and n_read > 1, interaction will not be returned as duplex group

return_collapsed == TRUE In case interaction has n_read > 1, interaction will be treated as duplex group

Value

`GInteractions` object with collapsed duplex groups

Examples

```
# load example of clustered data
data("RNADuplexesSampleData")
# some reads assigned to DG, some are not
table(is.na(RNADuplexSampleGI$dg_id))
# Return only DGs
gicollapsed <- collapse_duplex_groups(RNADuplexSampleGI, return_unclustered = FALSE)
# Return DGs and unclustered reads as well
gimixed <- collapse_duplex_groups(RNADuplexSampleGI, return_unclustered = TRUE)
```

```
# load small sample GInteractions and process it manually
data("RNADuplexesSmallGI")
# First, collapse duplicated reads. This adds n_reads and duplex ids
ginodup <- collapseIdenticalReads(SampleSmallGI)$gi_collapsed
# Second, run clustering, get DG ids
ginodup <- clusterDuplexGroups(ginodup)
# Return all DGs result in n=3 DGS, one of them formed by
# identical duplicated alignments
collapse_duplex_groups(ginodup, return_collapsed = TRUE)
# Return DGs, but drop duplicated returns n=2 DGs
collapse_duplex_groups(ginodup, return_collapsed = FALSE)
```

`col_check_rename` *Check the column names and types in read dataframe*

Description

Function to check the correct column names and types in dataframe input. Tries to guess the column names, if colnames are not provided, but the types are correct

Usage

```
col_check_rename(df, table_type = "STAR")
```

Arguments

`df`	input
`table_type`	one in c("STAR","bedpe")

Details

- Expected column names for bedpe file `c("chromA","startA",'endA',"chromB", 'startB','endB','readname`
- Expected colnames for STAR Chimeric junction input:
- For the 'old' chimeric detection scheme: `c("chr_donorA","brkpt_donorA","strand_donorA","chr_acceptor`
- For the 'new' chimeric detection scheme `c("chr_donorA","brkpt_donorA", "strand_donorA","chr_acceptor "brkpt_acceptorB","strand_acceptorB", "junction_type","repeat_left_lenA", "repeat_right_lenB "start_alnA","cigar_alnA", "start_alnB","cigar_alnB")`

Value

dataframe with the properly formatted columns

`compareMultipleInteractions` *Compare multiple RNA-RNA interactions sets*

Description

Combines all interaction into single superset by clustering & collapsing. Then compares every input entry with the superset. Overlaps between superset and inputs are recorded in a table as 0/1

Usage

```
compareMultipleInteractions(
  gi_samples_list,
  min_ratio = 0.3,
  minoverlap = 5,
  maxgap = 50,
  niter = 3,
  gi_superset = NULL,
  anno_gr = NULL
)
```

Arguments

`gi_samples_list`	anmes list with the `GInteractions` entries list('sample1'=gi1,'sample2'='gi2)
`min_ratio`	If the overlap-to-span ratio for either arm (A or B) for pair of chimeric reads is less than `min_arm_ratio`, then the total overlap for this pair is set to zero. Relevant to comparison of superset vs individual samples
`minoverlap`	Parameter for read clustering to create a superset. Minimum required overlap to for either arm (A or B) for pair of entries.
`maxgap`	Parameter for read clustering. Minimum required shift between start and end coordinates of arms for pair of overlapping entries.. If the shift is longer than `max_gap` for either arm, then total read overlap between those reads is zero.
`niter`	Internal parameter for debugging. Number of cluster& collapse iterations to find superset
`gi_superset`	Optional. Superset defining the space (all) of the interactions, against which inputs from the list will be compared.
`anno_gr`	Optional. `Granges` to annotate superset.

Value

dataframe recodding the overlaps between samples and supeset

Examples

```
# Create test set of RNA interactions
chrom <- "chr1"
start1 <- c(1, 11, 21, 31, 41, 51, 61, 71, 81, 91)
end1 <- start1 + 9
start2 <- c(101, 111, 121, 131, 141, 151, 161, 171, 181, 191)
```

```
end2 <- start2 + 9

anchor1 <- GRanges(seqnames = chrom, ranges = IRanges(start = start1, end = end1))
anchor2 <- GRanges(seqnames = chrom, ranges = IRanges(start = start2, end = end2))

interaction <- GInteractions(anchor1, anchor2)

# Ensure some overlaps
n <- length(interaction)
group_size <- ceiling(n / 2)
group_indices1 <- sort(sample(seq_len(n), group_size))
group_indices2 <- sort(sample(seq_len(n), group_size))
group_indices3 <- sort(sample(seq_len(n), group_size))

# Create separate GInteractions objects for each group
group1 <- interaction[group_indices1]
group2 <- interaction[group_indices2]
group3 <- interaction[group_indices3]

# format input and call comparison
a <- list("sample1" = group1, "sample2" = group2, "sample3" = group3)
res <- compareMultipleInteractions(a)
# comparison result
head(res$dt_upset)
# superset
res$gi_all
# dataframe for the Upset plot
res$dt_upset
```

---

computeGISelfOverlaps *Find overlaps between entries in* GInteractions

---

Description

Utility function to find overlapping reads in the input and calculate overlap scores. Removes self-hits. Computes overlap/span ratios for each interaction arm. Sum of the scores is recorded in 'weight' field

Usage

```
computeGISelfOverlaps(
  gi,
  id_column = "duplex_id",
  maxgap = 40,
  minoverlap = 10
)
```

Arguments

gi	input gi object
id_column	column which use for using as ids for entries
maxgap	parameter for call of `InteractionSet::findOverlaps()`
minoverlap	parameter for call `InteractionSet::findOverlaps()`

Value

dataframe with indexes of pairwise overlapsin input and columns for span, overlap, ratios of either read arm

Examples

```
data("RNADuplexesSmallGI")
computeGISelfOverlaps(SampleSmallGI)
```

`convert_gi_to_ranges` *Convert* `GInteractions` *object to* `Granges`

Description

Creates the 'long' `GRanges` by stacking the A and B arms one 'on top' of the other. Adds `id` and `group` fields as indicators of original index and interaction arm (A- left arm, B- right arm)

Usage

```
convert_gi_to_ranges(gi)
```

Arguments

`gi`	`GInteractions`

Value

`GRanges` twice the length of the input

Examples

```
data("RNADuplexesSmallGI")
convert_gi_to_ranges(SampleSmallGI)
```

`dd_get_chimeric_reads` *Accessor for* `chimeric_reads` *Slot*

Description

Retrieves the value of the `chimeric_reads` slot in a `DuplexDiscovererResults` object.

Usage

```
dd_get_chimeric_reads(object)

## S4 method for signature 'DuplexDiscovererResults'
dd_get_chimeric_reads(object)
```

Arguments

`object`	A `DuplexDiscovererResults` object.

Value

GInteractions object from the `chimeric_reads` slot.

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

---

dd_get_chimeric_reads_stats

Accessor for `chimeric_reads_stats` *Slot*

Description

Retrieves the value of the `chimeric_reads_stats` slot in a `DuplexDiscovererResults` object.

Usage

```
dd_get_chimeric_reads_stats(object)

## S4 method for signature 'DuplexDiscovererResults'
dd_get_chimeric_reads_stats(object)
```

Arguments

`object`	A `DuplexDiscovererResults` object.

Value

`tibble` from the `chimeric_reads_stats` slot.

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

`dd_get_duplex_groups` *Accessor for* `duplex_groups` *slot*

Description

Retrieves the value of the `duplex_groups` slot in a `DuplexDiscovererResults` object.

Usage

```
dd_get_duplex_groups(object)

## S4 method for signature 'DuplexDiscovererResults'
dd_get_duplex_groups(object)
```

Arguments

`object`	A `DuplexDiscovererResults` object.

Value

GInteractions object from the `duplex_groups` slot.

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

`dd_get_reads_classes` *Accessor for* `reads_classes` *Slot*

Description

Retrieves the value of the `reads_classes` slot in a `DuplexDiscovererResults` object.

Usage

```
dd_get_reads_classes(object)

## S4 method for signature 'DuplexDiscovererResults'
dd_get_reads_classes(object)
```

Arguments

`object`	A `DuplexDiscovererResults` object.

Value

`tibble` from the `reads_classes` slot.

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

dd_get_run_stats *Accessor for* run_stats *Slot*

Description

Retrieves the value of the run_stats slot in a DuplexDiscovererResults object.

Usage

```
dd_get_run_stats(object)

## S4 method for signature 'DuplexDiscovererResults'
dd_get_run_stats(object)
```

Arguments

object	A DuplexDiscovererResults object.

Value

tibble from the run_stats slot.

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

`drawGD,DuplexTrack-method`
Draw method for DuplexTrack

Description

`Gviz::AnnotationTrack` stacking algorithm is used to calculate vertical distribution of boxes for the interactions. Boxes coordinates are later imported for placing labels and arcs

Usage

```
## S4 method for signature 'DuplexTrack'
drawGD(GdObject, minBase, maxBase, prepare = FALSE, subset = TRUE, ...)
```

Value

pushes boxes, arcs and labels to viewport

`DuplexDiscovereR` *Analysis of the data from RNA duplex probing experiments*

Description

DuplexDiscovereR is a package for analysing data from RNA cross-linking and proximity ligation protocols such as SPLASH, PARIS, LIGR-seq and others, which provide information about intra-molecular RNA-RNA interactions through chimeric RNA-seq reads. Chimerically aligned fragments in these experiments correspond to the base-paired stretches (RNA duplexes) of RNA molecules . DuplexDiscovereR takes input in the form of chimericly or split -aligned reads, It implements procedures for alignment classification, filtering and efficient clustering of individual chimeric reads into duplex groups (DGs). Once DGs are found, RNA duplex formation and their hybridization energies are predicted. Additional metrics, such as p-values or mean DG alignment scores, can be calculated to rank and analyse the final set of RNA duplexes. Data from multiple experiments or replicates can be processed separately and further compared to check the reproducibility of the experimental method.

Details

DuplexDiscovereR

Author(s)

Egor Semenchenko

See Also

DuplexDiscovereR vignette

`DuplexDiscovererResults-class` *DuplexDiscovererResults*

Description

A helper S4 class to store the results of the full DuplexDiscovereR analysis. This class contains the following output:

- `duplex_groups`: clustered duplex groups.
- `chimeric_reads`: individual two-regions chimeric reads. Contains both clustered and unclustered reads. Clustered reads are linked to the duplex groups though 'dg_id' field in metadata
- `reads_classes`: dataframe parallel to the the input containing classification result and detected mapping type for each entry in the input
- `chimeric_reads_stats`: dataframe containing read type classification statistics
- `run_stats`: data frame containing statistics about the time and memory used by the pipeline

Usage

```
DuplexDiscovererResults(
  duplex_groups,
  chimeric_reads,
  reads_classes,
  chimeric_reads_stats,
  run_stats
)
```

Arguments

`duplex_groups`	**GInteractions** object with duplex groups
`chimeric_reads`	**GInteractions** object with chimeric reads
`reads_classes`	`tibble` (tbl_df) with read classification data.
`chimeric_reads_stats`	`tibble` (tbl_df) read type statistics.
`run_stats`	`tibble` (tbl_df) runtime and memory info

Details

Each output type has a corresponding accessor:

- `dd_get_duplex_groups()`
- `dd_get_chimeric_reads()`
- `dd_get_reads_classes()`
- `dd_get_chimeric_reads_stats()`
- `dd_get_run_stats()`

Value

A `DuplexDiscovererResults` object.

Slots

`duplex_groups` **GInteractions** object with duplex groups

`chimeric_reads` **GInteractions** object with chimeric reads

`reads_classes` `tibble` (tbl_df) with read classification data.

`chimeric_reads_stats` `tibble` (tbl_df) read type statistics.

`run_stats` `tibble` (tbl_df) runtime and memory info

See Also

`dd_get_duplex_groups()`, `dd_get_chimeric_reads()`, `dd_get_reads_classes()`, `dd_get_chimeric_reads_sta`
, `dd_get_run_stats()`

Examples

```
# load example input
data("RNADuplexesSmallGI")
data("RNADuplexesSampleData")
# run whole pipeline
result <- runDuplexDiscoverer(
    data = SampleSmallGI,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    sample_name = "run_example",
    lib_type = "SE",
    table_type = "STAR"
)
# access results
show(result)
gi_clusters <- dd_get_duplex_groups(result)
gi_reads <- dd_get_chimeric_reads(result)
df_reads <- dd_get_reads_classes(result)
dd_get_reads_classes(result)
dd_get_run_stats(result)
```

`DuplexTrack` *class for the visualization of RNA duplexes*

Description

Inherits the `Gviz::AnnotationTrack`, plots interaction ranges as boxes. Arguments from `Gviz::AnnotationTrack`, as `stacking` which set boxes layout are accepted. Parent aesthetics for labels are overwritten with Display parameters of this class. Accepts `GInteractions` object to plot and `GRanges` to define plot region

Duplexes which can be displayed on the plot range are connected with arcs. Duplexes which are partially outside of the range are displayed without arcs. Labeles and appearance can be controlled with display parameters

Arguments

`gi`	An `GInteractions` object
`gr_region`	`GRanges` region for plotting
`from`	Integer start coordinate of subset region. Used if `gr_region` is not provided
`to`	Integer end coordinate of subset region. Used if `gr_region` is not provided
`chromosome`	Chromosome of subset region. Used if `gr_region` is not provided
`strand`	Used if `gr_region` is not provided
`fill.column`	used for fill. Default is "" (empty) and triggers IGV color pallete. **Display parameters**

arcs.color Character. Color of the arcs. Default is "black".

arc.location Character in c('inner','outer','midpoint'). Location of the arcs in X axis relative to range. Default is "inner"

labels.v.offset.base Numeric. Base vertical offset for the labels. Default is 0.2. Other offesets are added to it.

labels.v.offset.trans Numeric. Vertical offset for trans labels. Applied when one part of the duplex is outside of the plot. Recommended ranges are in -0.5 to 0.5 Default is 0.0.

labels.h.offset.trans Numeric. Horizontal offset for trans labels. Applied when one part of the duplex is outside of the plot Value is in nucleotide units. Default is 0.0.

labels.v.offset.cis Numeric. Vertical offset for cis labels. Recommended ranges are in -0.5 to 0.5 Default is 0.0. Default is 0.0.

labels.h.offset.cis Numeric. Horizontal offset for cis labels. Value is in nucleotide units. Default is 0.0.

labels.fontsize Numeric. Font size of the labels. Default is 18.

label.cis.above Logical. Whether the cis labels should be above. When set to FALSE, labels are plot for each box separately. Default is TRUE

annotation.column1 Character. First annotation column to use for labels. Default is "group" and generated internally.

annotation.column2 Character. Second annotation column to use for labels. Default is "" (empty).

fill.column Character. Column used for fill. Default is "" (empty) and triggers IGV color pallete.

labels.color Character. Color of the labels. Default is 'black'.

labels.align Character. Alignment of the labels. Default is 'center'. Possible values are in c('left','right',center)

arcConstrain Numeric. Minimum gap distance between arms of the interaction to draw arcs

Examples

```
library(InteractionSet)
library(Gviz)
# generate input
anchor1 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(100, 600, 1100, 1600, 2100, 150, 400),
        end = c(200, 700, 1200, 1700, 2200, 250, 500)
    ),
    strand = "+"
)
anchor2 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(300, 800, 1300, 1800, 2300, 1500, 1700),
        end = c(400, 900, 1400, 1900, 2400, 1600, 1800)
    ),
    strand = "+"
)

interactions <- GInteractions(anchor1, anchor2, mode = "strict")
# define plotting range
gr_region <- range(anchor1, anchor2)
interactions$anno_A <- sample(LETTERS, length(interactions))
interactions$anno_B <- interactions$anno_A
a <- DuplexTrack(interactions, gr_region = gr_region, stacking = "dense")
plotTracks(a, stacking = "dense")
plotTracks(a, stacking = "squish", annotation.column1 = "anno_A")

# add interactions which are not fully in plot range: outside the range or on different chromosome()

# one left (A) interaction arm outside of the plot, other on different chromosome
new_anchor1 <- GRanges(
    seqnames = c("chr1", "chr2"),
    ranges = IRanges(
        start = c(10, 600),
        end = c(90, 700)
    ),
    strand = "+"
)
new_anchor2 <- GRanges(
    seqnames = c("chr1", "chr1"),
    ranges = IRanges(
        start = c(1500, 1000),
        end = c(1600, 1200)
    ),
    strand = "+"
)

new_interactions <- GInteractions(new_anchor1, new_anchor2)
new_interactions$anno_A <- c("A.out", "A.out_chr")
new_interactions$anno_B <- c("B.in", "B.in")
all_interactions <- c(interactions, new_interactions)

b <- DuplexDiscovereR::DuplexTrack(all_interactions,
    gr_region = gr_region,
    annotation.column1 = "anno_A",
    annotation.column2 = "anno_B"
)

plotTracks(b)

# to customize plot, one can call, to see options
DuplexDiscovereR::availableDisplayPars(b)
```

getChimericJunctionTypes *Classify chimeric junctions of two-arm reads into types*

Description

Chimeric reads which can be represented ans two-arm interactions can be divided into several categories based on the distance between the chimeric fragments and existence of the overlap between these fragments.

Usage

```
getChimericJunctionTypes(gi, normal_gap_threshold = 10)
```

Arguments

`gi`	`GInteractions` object
`normal_gap_threshold`	minimum allowed distance between chimeric arms

Details

Takes `GInteractions` object and classifies junctions into following categories

2arm normal chimeric read

2arm_short normal chimeric read with junction < *normal_gap_threshold*

self_ovl arms overlap

antisense_ovl arms overlap on the opposite strand

Value

gi object of the same size with the 'junction_type' field added

Examples

```
data("RNADuplexesSampleData")
preproc_df <- runDuplexDiscoPreproc(RNADuplexesRawBed, table_type = "bedpe")
preproc_gi <- makeGiFromDf(preproc_df)
preproc_gi <- getChimericJunctionTypes(preproc_gi)
table(preproc_gi$junction_type)
```

getRNAHybrids *Run prediciton of RNA hybridization*

Description

Calls RNAduplex from ViennaRNA to find base-pairs for every entry in the input, throws a message and system warning if it is not installed

Usage

```
getRNAHybrids(gi, fafile)
```

Arguments

`gi`	`Ginteraction` with pairs of regions
`fafile`	path to the .fasta file with genome

Value

object parallel to input with added energy GC content, dot-format base-pairings and lenghts of RNA hybrids will return the input, if RNAhybrids cannot be run

Examples

```
sequence <- paste0(
    "AGCUAGCGAUAGCUAGCAUCGUAGCAUCGAUCGUAAGCUAGCUAGCUAGCAUCGAUCGUAGCUAGCAUCGAU",
    "CGUAGCAUCGUAGCUAGCUAGCUAUGCGAUU"
)

# Save the sequence to a temp fasta file
fasta_file <- tempfile(fileext = ".fa")
chrom <- "test_chrA"
writeLines(c(">test_chrA", sequence), con = fasta_file)

# Create the GInteraction object
# Define start and end positions for the base-pairing regions
regions <- data.frame(
    start1 = c(1, 11, 21, 31, 41),
    end1 = c(10, 20, 30, 40, 50),
    start2 = c(91, 81, 71, 61, 51),
    end2 = c(100, 90, 80, 70, 60)
)
# GRanges objects for the anchors
anchor1 <- GRanges(seqnames = chrom, ranges = IRanges(start = regions$start1, end = regions$end1))
anchor2 <- GRanges(seqnames = chrom, ranges = IRanges(start = regions$start2, end = regions$end2))
interaction <- GInteractions(anchor1, anchor2)
# predict hybrids
# In case ViennaRNA is installed
## Not run:
gi_with_hybrids <- getRNAHybrids(interaction, fasta_file)

## End(Not run)
```

`getSpliceJunctionChimeras` *Identify chimeric junctions coinciding with the splice junctions*

Description

Marks interactions which starts/ends within specified shift from the known splice junctions.

Usage

```
getSpliceJunctionChimeras(
  gi,
  sj_gr,
  sj_tolerance = 20,
  sj_tolerance_strict = 10
)
```

Arguments

`gi`	**GInteractions** object
`sj_gr`	**Granges** object with the splice junctions data
`sj_tolerance`	maximum shift between either donor and acceptor splice sites and corresponding chimreic junction coordinates to count chimeric junction as splice junction
`sj_tolerance_strict`	maximum shift between either donor and acceptor splice sites irrespective of the particular splice junction. If both chimeric junction start and end correspond to donor or acceptor of any known junction, it is marked as splice junction. Used to catch novel combinations of known 3' and 5' sites

Value

gi object with added 'splicejnc' and field Additionally 'splicejnc_donor' 'splicejnc_acceptor' fields are added

Examples

```
data("RNADuplexesSampleData")
gi <- getSpliceJunctionChimeras(RNADuplexSampleGI, SampleSpliceJncGR)
table(gi$splicejnc)
table(gi$splicejnc_acceptor, gi$splicejnc_donor)
```

`get_arm_a` *Get left arm of GInteraction*

Description

Get left arm of GInteraction

Usage

```
get_arm_a(gi)
```

Arguments

`gi`	GInteractions object

Value

Granges object with the left (A) region

`get_arm_b` *Get right arm of GInteraction*

Description

Get right arm of GInteraction

Usage

```
get_arm_b(gi)
```

Arguments

`gi`	GInteractions object

Value

Granges object with the right (B) region

get_char_count_cigar *Count the length of the key type in CIGAR string*

Description

Takes CIGAR operands i.e M,N,S and sums the associated blocks length It is vectorized. i.e supports vector with CIGAR strings

Usage

```
get_char_count_cigar(strings, s)
```

Arguments

strings	CIGAR string vector
s	CIGAR operands

Value

vector with length values

Examples

```
# From a vector
get_char_count_cigar(c("4S18M22S", "25S26M"), "S")
get_char_count_cigar(c("18M22S", "20M20S"), "M")
```

get_chimeric_junctions_onestrand *Get chimeric junctions*

Description

Returns chimeric junction defined as range distance between the end and the start of the first and second range respectively

Usage

```
get_chimeric_junctions_onestrand(gi_intra)
```

Details

If the pair of interacting ranges is not on the same strand and chromosome, returns error

Value

Granges object with the

`get_colnames_and_types_for_input` *Get colnames for expected data types*

Description

Get colnames for expected data types

Usage

```
get_colnames_and_types_for_input(nameset)
```

Arguments

`nameset`	name of the table

Value

character vector

`makeDfFromGi` *Convert GInteractions to tibble*

Description

Converts `GInteractions` to tibble, preserves metadata

Usage

```
makeDfFromGi(gi)
```

Arguments

`gi`	GInteracttions

Details

Following naming conventions is used for region coordinates: `c('chromA','startA','endA','strandA','chromB','startB','endB','strandB')`

Value

tibble preserving metadata columns

See Also

`makeGiFromDf()`

Examples

```
data(RNADuplexesSmallGI)
converted_to_df <- makeDfFromGi(SampleSmallGI)
converted_to_gi <- makeGiFromDf(converted_to_df)
```

`makeGiFromDf` *Convert Dataframe to GInteractions*

Description

Converts dataframe-like object to the `GInteractions`.

Usage

```
makeGiFromDf(df)
```

Arguments

`df`	dataframe-like object. Should be convertable to tibble::tibble()

Details

arms will be consistent between different objects of same reference Following columns are looked up in input dataframe to parse region coordinates: c("chromA’,’startA’,’endA’,’strandA’,"chromB",’startB’,’endB’,’strandI `GInteractions(mode='strict')` is enforced, to ensure that the order of the regions Extra columns are stored as metadata fields

Value

GInteractions(mode=’strict’)

See Also

`makeDfFromGi()`

Examples

```
# load example GInteractions
data(RNADuplexesSmallGI)

converted_to_df <- makeDfFromGi(SampleSmallGI)
converted_to_gi <- makeGiFromDf(converted_to_df)
```

`preproc_chim_junction_out_pe`
Processing of of the STAR PE Chimeric.junction.out

Description

Calculates alignment coordinates and returns reads with categories

Usage

```
preproc_chim_junction_out_pe(dt, keep_all_columns = FALSE)
```

Arguments

`dt`	Chimeric.out.junction with the correct column names
`keep_all_columns`	• TRUE or FALSE. Keep CIGAR strings and junction coordinate columns

Details

#'

multimap multi-mapped read

multigap more than one junction (more than two 'N' in CIGAR string)

bad junction Artifacts. I.e alignments for both arms are continious, but with 'backward' chimeric junction was wrongly put

Value

tibble with annotated reads

`preproc_chim_junction_out_se`

Processing of of the STAR SE Chimeric.junction.out

Description

Calculates alignment coordinates and returns reads with categories

Usage

```
preproc_chim_junction_out_se(dt, keep_all_columns = FALSE)
```

Arguments

`dt`	Chimeric.out.junction with the correct column names
`keep_all_columns`	• TRUE or FALSE. Keep CIGAR strings and junction coordinate columns

Details

#'

multimap multi-mapped read

multigap more than one junction (more than two 'N' in CIGAR string)

bad junction Artifacts. I.e alignments for both arms are continious, but with 'backward' chimeric junction was wrongly put

Value

tibble with annotated reads

See Also

`col_check_rename()`

`preproc_generic` *Preprocess .bedpe input*

Description

Searches for the multi-mapped and bad reads (overlapping arms) Adds 'multimap', 'bad_junction' columns filled with 0 or 1 and 'multigap' = 0 for consistency with other pre-processing methods

Usage

```
preproc_generic(dt, keep_all_columns = TRUE)
```

Arguments

`dt`	dataframe with reads aligned to strictly two loci
`keep_all_columns`	keeep columns apart form the required from .bedpe format

Value

pre-processed dataframe

`preproc_generic_gi` *Preprocess GInteractions input*

Description

Searches for the multi-mapped reads (overlapping arms) Adds 'multimap', 'bad_junction' columns filled with 0/1 and 'multigap' = 0 for consistency with other pre-processing methods.

Usage

```
preproc_generic_gi(gi_raw, keep_all_columns = TRUE)
```

Arguments

`gi_raw`	`GInteractions` with inpit RNA interactions
`keep_all_columns`	keep columns apart from those required by .bedpe format

Value

pre-processed dataframe

`refresh_gi` *Refresh the* `GInteractions` *object*

Description

Sub-setting the `GInteractions` object does not reduce its `ranges` container For some applications, to save memory, we can safely reduce the size of the object by re-creating it. Also, it can be used to ensure the 'strict' mode of the regions in `ranges`

Usage

```
refresh_gi(gi)
```

Arguments

`gi`	`GInteractions` object

Value

`GInteractions` object with new ranges attribute

`RNADuplexesGeneCounts` *Gene counts on human chromosome 22, embryonic stem cells*

Description

File generated by mapping with STAR using `--quantMode GeneCounts` see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details on the pre-processing and sub-setting the

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1445 rows and 2 columns.

Value

`tibble` with columns of Chimeric.junction.out

Source

SequenceReadArcive

`RNADuplexesRawBed` *Chimeric reads of SPLASH converted to .bedpe fromat*

Description

A Chimeric.out.Junction file with a subset of chr 22 Chimeric reads detected by SPLASH protocol in Human embryonic stem cells.

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 2040 rows and 10 columns.

Value

`tibble` with columns of bedpe format

Source

SequenceReadArcive Reads were aligned with STAR and filtered to contain only reads which could be represented as 2-arm chimeric alignments. Converted to the bedpe format see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details on the pre-processing and sub-setting the data

`RNADuplexesRawChimSTAR` *Chimeric reads of SPLASH*

Description

A Chimeric.out.Junction file with a subset of chr 22 Chimeric reads detected by SPLASH protocol in Human embryonic stem cells.

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 5000 rows and 21 columns.

Value

`tibble` with columns of Chimeric.junction.out

Source

SequenceReadArcive Reads were aligned with STAR see `system.file("extdata/scripts", "DD_data_generation.` `package = "DuplexDiscovereR")` for details on the pre-processing and sub-setting the data

---

`RNADuplexSampleClustReads`
RNA duplex reads of SPLASH, clustered and assigned to duplex groups

---

Description

`GInteractions` read-level object containing processed reads,annotated with duplex group ids, read types gene names and p-values

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `StrictGInteractions` of length 2090.

Value

`GInteractions` with

- `n_reads_dg` : number of reads in the duplex group (DG)
- `duplex_id` : temporary id for RNA duplexes which could be found before clustering (duplicated or shifted by couple of nt)
- `dg_id` :id of the duplex group
- `score` : median alignment score in duplex group
- other columns inherited from the STAR Chimeric.out.Junction

Source

SequenceReadArcive Reads were aligned with STAR and duplex groups were identified see `system.file("extdata/sc` `"DD_data_generation.R", package = "DuplexDiscovereR")` for details on the data generation proccedure.

RNADuplexSampleDGs *RNA duplex reads of SPLASH, clustered and collapsed to duplex groups*

Description

`GInteractions` duplex group -level object containing detected duplex groups, annotated with duplex group ids, gene_names and p-values

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `StrictGInteractions` of length 79.

Value

`GInteractions` with

- `n_reads` : number of reads in the duplex group (DG)
- `dg_id` :id of the duplex group
- `p_val` : BH adjusted p-value of testing to reject hypothesis of DG arising from random ligation
- `score` : median alignment score in duplex group
- other columns with `.A` and `.B` annotating to which genes either arm of the DG maps

Source

SequenceReadArcive Reads were aligned with STAR and duplex groups were identified see `system.file("extdata/sc` `"DD_data_generation.R", package = "DuplexDiscovereR")` for details on the data generation procedure.

RNADuplexSampleGI *RNA duplex reads of SPLASH derived from chimeric alignments*

Description

`GInteractions` read-level object containing two-arm chimeric reads extracted from mapping output and which can be represented in the `GInteraction` object

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `StrictGInteractions` of length 2090.

Details

see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details on the data generation proccedure.

Value

`GInteractions` with

- `readname` : read name
- `map_type` : type of the mapped read (2arm by design of pre-filtering)
- `junction_type` : if read jucntion is too short, or it not a 'true' ligated reads because of the jucntoin coincides with splice junction
- cigar_aln* columns inherited from the STAR Chimeric.out.Junction output

Source

SequenceReadArcive

`runDuplexDiscoPreproc` *Run pre-processing of chimeric reads input*

Description

Imports dataframe with reads (*.bedpe* or *Chimeric.out.junction*) or `GInteractions` object. Checks column names or tries to quess them if not provided. Adds necessary annotation depending on the input type, For *STAR* input, calculates length of the alignments and marks unique 2-arm alignments. For the *.bedpe* or `GInteractions` input, all entries are already represented as reads with two different aligned parts (2-arm), so only check for unique readname is performed.

Usage

```
runDuplexDiscoPreproc(
  data,
  table_type,
  library_type = "SE",
  keep_metadata = TRUE,
  return_gi = FALSE,
  min_arm_len = 15
)
```

Arguments

`data`	Either dataframe-like object: *Chimeric.out.junction* from *STAR* or *.bedpe* - formatted or `GInteractions` object from **InteractionSet** package
`table_type`	in `c("STAR","bedpe")` for Chimeric.out.Junction or generic input
`library_type`	`c("SE","PE")` for pair- or single- end input
`keep_metadata`	`c(TRUE,FALSE)` Whether extra fields like CIGAR strings and junction coordinates should be kept
`return_gi`	if the return object should be `GInteractions`
`min_arm_len`	minimum allowed length of the alignment arm. Read will be dropped if either arm is shorter

Details

If not existed, adds fields required for the downstream steps: 'readname', 'map_type', 'score', 'n_reads'. 'map_type' field determines the type of the chimeric read:

multimap multi-mapped read

multigap more than one junction (more than two 'N' in CIGAR string)

bad junction Artifacts or possibly unaccounted types. I.e alignments for both arms are continuous, but with 'backward' chimeric junction was wrongly introduced in the mapping

Value

tibble with new metadata fields OR GInteractions if `return_gi` is set to TRUE

Examples

```
# load data
data(RNADuplexesSampleData)
# with bedpe input
preproc_reads <- runDuplexDiscoPreproc(RNADuplexesRawBed, table_type = "bedpe")
# with STAR input
preproc_reads_star <- runDuplexDiscoPreproc(RNADuplexesRawChimSTAR,
    table_type = "STAR",
    keep_metadata = FALSE
)
```

`runDuplexDiscoverer` *Executes all steps of DuplexDiscovereR pipeline*

Description

Generates GInteractions object with duplex groups from the STAR Chimeric.out.junction or bedpe file. Classifies reads, annotates reads by overlap with the gene or transcript features, calculates p-values and hybridization energies. Additionally, returns mappings from duplex groupd back to genes.

Usage

```
runDuplexDiscoverer(
  data,
  table_type = "",
  junctions_gr = NULL,
  anno_gr = NULL,
  anno_gr_keys = c("gene_id", "gene_name", "gene_type"),
  fafile = NULL,
  df_counts = NULL,
  sample_name = "sample",
  lib_type = "SE",
  min_junction_len = 5,
  max_gap = 50,
  min_arm_ratio = 0.1,
  min_overlap = 10,
  max_sj_shift = 10,
  gap_collapse_similar = 2,
  collapse_n_inter = 5,
  trim_alignments = FALSE,
  trim_length = 40,
  min_arm_len = 9,
  compute_p_values = TRUE
)
```

Arguments

`data`	dataframe-like object with the split reads. Output of Chimeric.out.junction or dataframe with fileds defined by bedpe format: c("chromA","startA",’endA’,"chromB",’startB’,’endB ...) Alternatively, `GInteractions` object
`table_type`	one in c("STAR","bedpe") Defines the type of the input dataframe. ignored if input data is `GInteractions`
`junctions_gr`	**GRanges** object with the splice junction coordinates
`anno_gr`	**GRanges** object to use for the annotation of the interactions. Optional
`anno_gr_keys`	c() vector with names of metadata fields in anno_gr which will be used for the annotation. Argument passed to `annotateGI()` function. The c(’gene_id’,’gene_name’,’gene_type’) columns in anno_gr are used by default.
`fafile`	path to the genome .fasta file. Used to calculate hybridization energy with *RNADuplex*. Sequence names should correspond to the sequences from which the mapping index was created. Optional
`df_counts`	A two- column dataframe with counts. Counts are used for p-value calculation. The first column should match the ’gene_id’ feature in anno_gr. The second column is the respective count. Optional
`sample_name`	A name of the sample, used for assembling the analysis statistics dataframe
`lib_type`	one in c(’SE’,’PE’). Type of the seqeuncing library. Default is ’SE’
`min_junction_len`	a minimum allowed distance between chimeric arms for the read input. Reads with the junction closer than `min_junction_len` are annotated as ’2arm_shot’ and not clustered to duplex groups
`max_gap`	Parameter for read clustering. Minimum required shift between start and end coordinates of arms for pair of overlapping chimeric reads. If the shift is longer than `max_gap` for either arm, then total read overlap between those reads is zero.
`min_arm_ratio`	Parameter for read clustering. If the overlap-to-span ratio for either arm (A or B) for pair of chimeric reads is less than `min_arm_ratio`, then the total overlap for this pair is set to zero.
`min_overlap`	Parameter for read clustering. Minimum required overlap to for either arm (A or B) for pair of chimeric reads.
`max_sj_shift`	Maximum shift between either donor and acceptor splice sites and chimeric junction coordinates to count chimeric junction as splice junction
`gap_collapse_similar`	Parameter for read clustering (iterative step). Analogous to the max_gap, but applied `collapse_n_inter` times during the iterative merging step. Reduce this to 1 or 2 to lower RAM usage for clustering the library with many similar reads.

`collapse_n_inter`	Parameter for read clustering (iterative step). Number of iterations to repeat step of collapsing of the highly similar chimeric reads. Increasing this from i.e 0 to 5 reduces clustering time and memory for the libraries with many overlapping reads.
`trim_alignments`	TRUE or FALSE. Whether to trim arms alignments to 'trim_length' nucleotide around chimeric junction
`trim_length`	target size of trimmed alignment
`min_arm_len`	minimum allowed length of the alignment arm. Read will be dropped if either arm is shorter
`compute_p_values`	TRUE or FALSE. whether to calcualte random ligation test

Details

This is a main function to do the initial discovery of the RNA duplexes after the chimeric read mapping. It wraps following procedures:

- Classifies the input reads by the mapping type. Keeps 2-arm chimeric reads for downstream analysis
- Compares 2arm duplex reads against provided splice junctions
- Classifies 2arm duplexes into spurious self-overlapping, splice junction categoris
- Performs clustering of the remaining reads into duplex groups
  - Collapses identically mapped reads
  - Collapses closely located reads, almost identical reads
  - Finds duplex groups throughout whole data set
- Annotates duplex groups with genomic features if annotation is provided
- Calculates p-values if gene counts and annotation are provided
- Calculates hybridization energies if path to the .fasta file is provided

Value

a list with the following keys

`duplex_groups` `GInteractions` object with chimeric reads clustered duplex groups

`chimeric_reads` `GInteractions` object with non-collapsed chimeric reads

`reads_classes` `tbl_df` dataframe parallel to the the input dataframe, annotated with read categories and duplex groups

`chimeric_reads_stats` `tbl_df` dataframe containing read type classification statistics

`run_stats` `tbl_df` dataframe with the time and memory info about the run

See Also

`DuplexDiscovererResults()`

Examples

```
library(DuplexDiscovereR)
# load data
data("RNADuplexesSampleData")
result <- runDuplexDiscoverer(
    data = RNADuplexesRawChimSTAR,
    junctions_gr = SampleSpliceJncGR,
    anno_gr = SampleGeneAnnoGR,
    df_counts = RNADuplexesGeneCounts,
    sample_name = "test clustering",
    fafile = NULL,
    collapse_n_inter = 3,
    lib_type = "SE",
    table_type = "STAR"
)
# see results object
print(result)
# duplex groups
dd_get_duplex_groups(result)
# individual chimeric reads
dd_get_chimeric_reads(result)
# counts of detected read tyoes
dd_get_chimeric_reads_stats(result)
```

`SampleGeneAnnoGR` *Gene coordinates on human chromosome 22*

Description

Granges containing gene coordinates of human chromosome 22 obtained from GENCODEv44 annotaion

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `GRanges` of length 1445.

Details

see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details

Value

statdatd GENCODE gtf fields

Source

GENCODEv44

`SampleSmallGI` *RNA duplex reads of SPLASH derived from chimeric alignments*

Description

`GInteractions` object containing two-arm chimeric reads extracted from mapping output and which can be represented in the `GInteraction` object and subset to chr22: 23877144-45562960 '*'

Usage

```
data(RNADuplexesSmallGI)
```

Format

An object of class `StrictGInteractions` of length 14.

Details

see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details on the data generation procedure.

Source

SequenceReadArcive

`SampleSpliceJncGR` *Gene coordinates on human chromosome 22*

Description

`Granges` containing coordinates of splice junctions human chromosome 22 obtained from GENCODEv44 annotaion

Usage

```
data(RNADuplexesSampleData)
```

Format

An object of class `GRanges` of length 8465.

Details

see `system.file("extdata/scripts", "DD_data_generation.R", package = "DuplexDiscovereR")` for details

Value

statdatd GENCODE gtf fields

Source

GENCODEv44

`show,DuplexDiscovererResults-method`
Show Method for DuplexDiscovererResults class

Description

This method provides a summary of the DuplexDiscovererResults object. It prints `chimeric_reads_stats` followed by the `run_stats`.

Usage

```
## S4 method for signature 'DuplexDiscovererResults'
show(object)
```

Arguments

`object`	A `DuplexDiscovererResults` object.

Value

None. Prints a formatted summary.

`show,DuplexTrack-method`
Show method for DuplexTrack

Description

Show method for DuplexTrack

Usage

```
## S4 method for signature 'DuplexTrack'
show(object)
```

Arguments

`object`	DuplexTrack.

Value

class representation

Examples

```
library(InteractionSet)
anchor1 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(100, 600, 1100, 1600, 2100),
        end = c(200, 700, 1200, 1700, 2200)
    ),
    strand = "+"
)
anchor2 <- GRanges(
    seqnames = "chr1",
    ranges = IRanges(
        start = c(300, 800, 1300, 1800, 2300),
        end = c(400, 900, 1400, 1900, 2400)
    ),
    strand = "+"
)

interactions <- GInteractions(anchor1, anchor2, mode = "strict")
gr_region <- range(anchor1, anchor2)
a <- DuplexTrack(interactions, gr_region = gr_region, stacking = "dense")
show(a)
```

`subset_gi` *Subset the* `GInteractions` *object to single interaction*

Description

Sub-setting the `GInteractions` object does not reduce its `ranges` container This function selects `Ginteraction` by index and reduces the `ranges`

Usage

```
subset_gi(gi, k)
```

Arguments

`gi`	`GInteractions` object

Value

`GInteractions` with the range atribure reduced to single interaction

`trimAroundJunction` *Extract regions around chimeric junction*

Description

Trim alignements to contain only 'extract len' nucleotides adajcent to the chimeric junction

Usage

```
trimAroundJunction(dt, extract_len = 30)
```

Arguments

`dt`	table with the
`extract_len`	the length of the seqeunce to trim

Details

In case of the long alignemtns, it may be necessary trim chimeric alignments to identify RNA duplex. If 'extract_len' is longer than the read alignemnt length, then no trimming is performed

Value

dataframe with the trimmed alignments

Examples

```
data("RNADuplexesSampleData")
dt_preproc <- runDuplexDiscoPreproc(RNADuplexesRawChimSTAR,
    table_type = "STAR", library_type = "SE"
)
trimAroundJunction(dt_preproc, 40)
```

`writeGiToSAMfile` *Write reads to sam file*

Description

Writes interactions to the sam file for visualization in extrnal browsers. Takes input as GInteractions object containing reads or duplex groups.

Usage

```
writeGiToSAMfile(
  gi_coords,
  file_out,
  distance_chim_junction = 10000,
  read_name_column = "readname",
  id_column = "dg_id",
  genome = "",
  sample_name = "noname_sample"
)
```

Arguments

`gi_coords`	input Ginteraction object
`file_out`	path to write output file
`distance_chim_junction`	maximum distance between input duplex groups/reads, which will be represented as the single-line in .sam file. Junction will be output as N- gap. For the interactions with longer distances, chimeric junction will be represented as MR:Z:i tag
`read_name_column`	character field, pointing out to read names. Read names are generated automatically if not provided.
`id_column`	character name of the field containing integer duplex group ids. NA are replaced with zeros
`genome`	character. Genome version. Required for the retrieval of sequence lengths for sam file header- SQ and SN tags. For convenience, hg38 and hg19 chromosome lengths will be assigned automatically. If the value is not in c('hg38','hg19'), seqlengths will be looked for be in attribute in seqlengths() of regions(gi_coords)
`sample_name`	name to use in RG SAM tag in header

Value

no object is returned

Examples

```
# Load test data
data("RNADuplexesSampleData")
# if the input is read-based, it should have integer duplex group ids
# here, we have 2090 reads
length(RNADuplexSampleGI)
# among them 300 reads does not belong to any DG
# missing ids will be converted to 0
table(is.na(RNADuplexSampleGI$dg_id))
tmpf <- tempfile(".sam")
writeGiToSAMfile(
    gi_coords = RNADuplexSampleGI,
    id_column = "dg_id",
    file_out = tmpf,
    distance_chim_junction = 1e5,
    genome = "hg38"
)
```

Index