Package ‘RgnTX’

May 22, 2024

Title  Colocalization analysis of transcriptome elements in the presence of isoform heterogeneity and ambiguity

Version  1.6.0

Description  RgnTX allows the integration of transcriptome annotations so as to model the complex alternative splicing patterns. It supports the testing of transcriptome elements without clear isoform association, which is often the real scenario due to technical limitations. It involves functions that do permutation test for evaluating association between features and transcriptome regions.

License  Artistic-2.0

Encoding  UTF-8

Roxygen  list(markdown = TRUE)

RoxygenNote  7.2.0

biocViews  AlternativeSplicing, Sequencing, RNASeq, MethylSeq, Transcription, SplicedAlignment

Imports  GenomeInfoDb, GenomicFeatures, GenomicRanges, ggplot2, graphics, IRanges, methods, regioneR, S4Vectors, stats, TxDb.Hsapiens.UCSC.hg19.knownGene

Depends  R (>= 4.2.0)

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

VignetteBuilder  knitr

Config/testthat/edition  3

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

git_branch  RELEASE_3_19

git_last_commit  b54e1c6

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-05-21

Author  Yue Wang [aut, cre], Jia Meng [aut]

Maintainer  Yue Wang <yue.wang19@student.xjtlu.edu.cn>
calculateShift

Calculate positional shifting over transcriptome

Description

The first step of calculating positional shift over transcriptome regions.

Usage

calculateShift(regions, disp, direction = "right", strand = "+")
calculateShift

Arguments

- **regions**: A feature set, which should be a GRangesList object.
- **disp**: A data frame object. It should have three columns, which are start: starting positions. Each value represents a starting position in each input feature; width: widths. Each value represents a width of each region to be picked from each feature; names: corresponding transcript ids.
- **direction**: Either to be character "left" or "right", which means the direction to which the starting position is shifting. The former means moving to the direction of 5' while the latter means moving to 3'.
- **strand**: Either to be "+" or "-".

Value

A GRanges object.

See Also

- `extractRegions`

Examples

```r
# Take five transcripts.
# Extract the last 200 nt regions from their CDS part.
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
trans.id.pstv <- c("170", "782", "974", "1364", "1387")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

# Download the CDS part of all transcriptome
cds.tx0 <- cdsBy(txdb, use.names = FALSE)

# pick the CDS part of these five transcripts
cds.p <- cds.tx0[trans.id.pstv]

width <- 200
disp.p.l <- data.frame(
  start = as.numeric(max(end(cds.p))),
  distance = width - 1,
  names = trans.id.pstv
)
R.p.l <- calculateShift(
  regions = cds.p, disp = disp.p.l,
  direction = "left", strand = "+")
```
Evaluation function

This function calculates the mean of the distance from each region of set RS1 to the closest region in RS2.

Usage

distanceTx(A, B, beta = 0.2, ...)

Arguments

A Region set 1. A Granges or GRangesList object.
B Region set 2. A Granges or GRangesList object.
beta It is a user-defined argument that can filter out the corresponding percent of largest distance values. Default value is 0.2.
... Any additional parameters needed.

Value

A numeric object.

See Also

overlapWidthTx, overlapCountsTx

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
A <- randomizeTx(
    txdb, trans.ids,
    random_num = 20,
    random_length = 100
)
B <- randomizeTx(
    txdb, trans.ids,
    random_num = 20,
    random_length = 100
)
distanceTx(A, B, beta = 0.2)
extractRegions

**Extract regions**

**Description**

This function receives three arguments: the scope region set, the target region set and the type of strand. It returns a subset of target region set, which is the intersection of the target region set and the scope region set.

**Usage**

```r
extractRegions(regions_A, R, strand = "+")
```

**Arguments**

- `regions_A`  
  The scope region set. A GRangesList object. The name of each list element should be the transcript id that it pertains to.
- `R`  
  The target region set. A GRanges object.
- `strand`  
  The strand type of the transcripts. It has options "+" and "-".

**Value**

A GRangesList object.

**See Also**

- `calculateShift`

**Examples**

```r
# Take five transcripts.  
# Extract the last 200 nt regions from their CDS part.  
library(TxDb.Hsapiens.UCSC.hg19.knownGene)  
trans.id.pstv <- c("170", "782", "974", "1364", "1387")  
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

cds.tx0 <- cdsBy(txdb, use.names = FALSE)

cds.p <- cds.tx0[trans.id.pstv]

width <- 200  
disp.p.1 <- data.frame(  
  start = as.numeric(max(end(cds.p))),  
  distance = width - 1,  
  names = trans.id.pstv)
```


getPermSpaceByFeatures

Description

This function returns a default permutation space for features with isoform ambiguity. The default permutation space of a feature is the aggregate of the multiple transcripts it may overlap with. It requires the input feature to be GRanges format.

Usage

getPermSpaceByFeatures(features, txdb, type = "mature")

Arguments

  - **features**: A GRanges object.
  - **txdb**: A TxDb object.
  - **type**: A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of regions over transcriptome.

getFormatCorrect

Description

This function makes sure the two input region sets are in the correct format required by RgnTX evaluation functions.

Usage

getFormatCorrect(A, B)

Arguments

  - **A**: Region set 1. A Granges or GRangesList object.
  - **B**: Region set 2. A Granges or GRangesList object.

Value

  A list object.

R.p.l <- calculateShift(regions = cds.p, disp = disp.p.l, direction = "left", strand = "+")
R.cds.last200 <- extractRegions(regions_A = cds.p, R = R.p.l, strand = "+")
getPermSpaceByTxID

Value

A list object, which contains two elements.

• perm.space: A GRangesList object that includes all the transcripts input features may overlap with.
• index: It contains a series of numbers indicating which feature these transcripts are respectively associated with.

See Also

getPermSpaceByTxID, getPermSpaceByType

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
taxdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
permSpace <- getPermSpaceByFeatures(features = m6A_sites_data[1:100], txdb)

getPermSpaceByTxID Get permutation space by specifying transcript ids

Description

This function returns 5'UTR/CDS/3'UTR/mRNA/full part of transcriptome regions grouped by corresponding transcript ids.

Usage

getPermSpaceByTxID(trans_ids = "all", txdb, type = "mature")

Arguments

trans_ids A character object. The transcript ids. Default is "all". If it takes the default value "all", the space that users get will be the whole transcriptome.

txdb A TxDb object.

type A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.

Value

A GRangesList object.

See Also

getPermSpaceByType, getPermSpaceByFeatures
getPermSpaceByType

Examples

trans.ids <- c("170", "782", "974", "1364", "1387")
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
permspace <- getPermSpaceByTxID(trans.ids, txdb)

getPermSpaceByType <- getPermSpaceByType(txdb, type = "mature")

Arguments

- **txdb**: A TxDb object.
- **type**: A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.

Value

A GRangesList object.

See Also

getPermSpaceByTxID, getPermSpaceByFeatures

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
permSpace <- getPermSpaceByType(txdb, type = "CDS")
### getPvalZscore

**Description**
Calculate a p-value and z-score based on observed value and random evaluation values.

**Usage**
```r
getPvalZscore(orig.ev, rand.ev, pval_z = FALSE)
```

**Arguments**
- `orig.ev`: Observed value.
- `rand.ev`: Random evaluation values.
- `pval_z`: Boolean. Default is FALSE. If FALSE, the p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, the p-value is calculated based on a z-test.

**Value**
A p-value and a z-score.

### getStopCodon

**Description**
Get stop codon regions for input transcripts. This is an example of customPick function.

**Usage**
```r
getStopCodon(trans_ids, txdb, ...)
```

**Arguments**
- `trans_ids`: A character object containing transcript ids.
- `txdb`: A TxDb object.
- `...`: Any additional parameters needed.

**Value**
A numeric object.
Examples

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
RS2 <- getStopCodon(trans.ids, txdb)
```

---

**getTransInfo**

*Get transcript information*

**Description**

Generate a data frame object that contains information about input genomic feature set and its mapping results over the transcriptome.

**Usage**

```r
getTransInfo(A, txdb)
```

**Arguments**

- `A`: Genomic feature set, which should be a `GRanges` object.
- `txdb`: A TxDb object.

**Value**

A `data.frame` object containing the following components:

- `index_trans`: The label of transcripts.
- `index_features`: The label of genomic features.
- `seqnames`: The chr name.
- `features_pos`: The starting coordinate of each genomic feature.
- `width_features`: The width of each genomic feature.
- `strand`: The strand type of each genomic feature.
- `trans_ID`: The ids of the transcripts that each feature can be mapped to.

**Examples**

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
getTransInfo(A = m6A_sites_data[1:100], txdb)
```
**GRanges2GRangesList**  
*Convert a GRanges object to a GRangesList object*

**Description**

Convert a GRanges object to a GRangesList object. The output region set follows the format required by the main permutation test functions.

**Usage**

```r
GRanges2GRangesList(A = NULL)
```

**Arguments**

- `A`  
  A GRanges object.

**Details**

If input GRanges object has a metadata named as "group", ranges having the same group number represent a region. If not, a range is a region. A region in the input set will be outputted as a list element IN returned GRangesList object.

**Value**

A GRangesList object.

**See Also**

`GRanges2GRangesList`

**Examples**

```r
library(GenomicRanges)
GRanges.object <- GRanges(
  Rle(c("chr2", "chr2", "chr1", "chr3")),
  IRanges(1:4, width = 5)
)
# Assign the first and the second ranges to the same element.
GRanges.object$group <- c(1, 1, 2, 3)
GRangesList.object <- GRanges2GRangesList(GRanges.object)
```
## GRangesList2GRanges

**Convert a GRangesList object to a GRanges object**

### Description

Convert a GRangesList object to a GRanges object. The output region set follows the format required by the RgnTX permutation test functions, which should have metadata columns 'group' and 'transcriptsHits'.

### Usage

```r
GRangesList2GRanges(A = NULL)
```

### Arguments

- `A` A GRangesList object.

### Value

A GRanges object. Its transcript ids (if available) should be contained in a metadata column named "transcriptsHits", which are provided by the names of input GRangesList object.

### See Also

- `GRanges2GRangesList`

### Examples

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
RS1 <- randomizeTx(txdb, trans.ids, random_num = 100, random_length = 100)
RS1 <- GRangesList2GRanges(RS1)
```

---

## overlapCountsTx

**Evaluation function**

### Description

This function receives two region sets and returns the number of their overlaps.

### Usage

```r
overlapCountsTx(A, B, count_once = TRUE, over_trans = TRUE, ...)
```
Arguments

A Region set 1. A \texttt{GRangesList} object.
B Region set 2. A \texttt{GRangesList} object.
count\_once Whether the overlap of multiple B regions with a single A region should be counted once or multiple times.
over\_trans Whether the overlapping is counted over the transcriptome or over the genome.
...

Value

A numeric object.

See Also

\texttt{overlapCountsTx}

Examples

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
exons.tx0 <- exonsBy(txdb)
regions.A <- exons.tx0[trans.ids]
A <- randomizeTransByOrder(regions.A, random_length = 200)
B <- randomizeTransByOrder(regions.A, random_length = 200)
overlapCountsTx(A, B)
```

Description

Evaluation function. This function receives a feature set (with isoform ambiguity) and a transcriptome region set (without isoform ambiguity), and returns a weighted number of overlaps between them.

Usage

\texttt{overlapCountsTxIA(A, B, ...)}

Arguments

A A feature set, which should be \texttt{GRanges}.
B A region set, which should be \texttt{GRangesList}.
... Any additional parameters needed.
Value

A numeric object.

See Also

overlapWidthTx, distanceTx, overlapCountsTx

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
RS1 <- m6A_sites_data[1:100]

trans.info <- getTransInfo(RS1, txdb)
trans.ids <- trans.info[, "trans_ID"]
RS2 <- getStopCodon(trans.ids, txdb = txdb)

# Evaluation step.
orig.ev <- overlapCountsTxIA(RS1, RS2)

overlapWidthTx(A, B, ...)

Arguments

A Region set 1. A Granges or GRangesList object.
B Region set 2. A Granges or GRangesList object.
... Any additional parameters needed.

Value

A numeric object.

See Also

overlapCountsTx, distanceTx
**Examples**

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
A <- randomizeTx(
  txdb, trans.ids, random_num = 20,
  random_length = 100
)
B <- randomizeTx(
  txdb, trans.ids = trans.ids, random_num = 20,
  random_length = 100
)
overlapWidthTx(A, B)
```

---

**permTestTx**

**Perform permutation test**

**Description**

Perform permutation test for evaluating spatial association between a feature set and a region set.

**Usage**

```r
permTestTx(RS1 = NULL, RS2 = NULL, txdb = NULL, type = "mature",
ntimes = 50, ev_function_1 = overlapCountsTx, ev_function_2 = overlapCountsTx,
pval_z = FALSE, ...)
```

**Arguments**

- **RS1**
  - The region set to be randomized. It should be in the GRanges or GRangesList format.
- **RS2**
  - The region set to be compared with. It should be in the GRanges or GRangesList format.
- **txdb**
  - A TxDb object.
- **type**
  - A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.
- **ntimes**
  - Randomization times.
- **ev_function_1**
  - Evaluation function defines what statistic to be tested between RS1 and RS2. Default is overlapCountsTx.
- **ev_function_2**
  - Evaluation function defines what statistic to be tested between each element in RSL and RS2. Default is overlapCountsTx.
- **pval_z**
  - Boolean. Default is FALSE. If FALSE, the p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, the p-value is calculated based on a z-test.
- **...**
  - Any additional parameters needed.
Details

permTestTxIA only needs users to input two region sets. It will automatically randomize the first region set into transcriptome.

Value

A list object, which is defined to be permTestTx.results class. It contains the following items:

- RSL: Randomized region sets of RS1.
- RS1: The feature set to be randomized.
- RS2: The region set to be compared with the feature set.
- orig.ev: The value of overlapping counts between RS1 and RS2.
- rand.ev: The values of overlapping counts between each element in RSL and RS2.
- pval: p-value of the test.
- zscore: Standard score of the test.

See Also

plotPermResults

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
exons.tx0 <- exonsBy(txdb)
trans.ids <- sample(names(exons.tx0), 500)

A <- randomizeTx(txdb, trans.ids, random_num = 100, random_length = 100)
B <- c(randomizeTx(txdb, trans.ids, random_num = 75, random_length = 100), A[1:25])

permTestTx_results <- permTestTx(A, B, ntimes = 5)

permTestTxIA

Perform permutation test

Description

Perform permutation test for evaluating spatial association between some features (with isoform ambiguity) and a region set. It randomizes the features and compares it with the region set to see if there is an association between the features and the region set. The difference between this function and permTestTx is that it is for RNA-related genomic features that have isoform ambiguity, i.e., features that one does not know which transcript they come from.
permTestTxIA

Usage

permTestTxIA(RS1 = NULL,
RS2 = NULL,
txdb = NULL,
type = 'mature',
ntimes = 50,
ev_function_1 = overlapCountsTx,
ev_function_2 = overlapCountsTx,
pval_z = FALSE,
...)

Arguments

RS1 The feature set to be randomized. It should be in the GRanges or GRangesList format.
RS2 The region set to be compared with. It should be in the GRanges or GRangesList format.
txdb A TxDb object.
type A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.
ntimes Randomization times.
ev_function_1 Evaluation function defines what statistic to be tested between RS1 and RS2. Default is overlapCountsTx.
ev_function_2 Evaluation function defines what statistic to be tested between each element in RSL and RS2. Default is overlapCountsTx.
pval_z Boolean. Default is FALSE. If FALSE, the p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, the p-value is calculated based on a z-test.
...
Any additional parameters needed.

Details

permTestTxIA only needs users to input two region sets. It will automatically randomize the first region set into transcriptome.

Value

A list object, which is defined to be permTestTxIA.results class. It contains the following items:

• RSL: Randomized region sets of RS1.
• RS1: The feature set to be randomized.
• RS2: The region set to be compared with the feature set.
• orig.ev: The value of overlapping counts between RS1 and RS2.
• rand.ev: The values of overlapping counts between each element in RSL and RS2.
• pval: p-value of the test.
• zscore: Standard score of the test.
Perform permutation test for evaluating spatial association between RNA features and a specified kind of regions. The latter is defined by the customPick_function argument input by users. The difference between this function and `permTestTx_customPick` is that it is for RNA-related genomic features that have isoform ambiguity, i.e., features that one does not know which transcript they comes from.

### Usage

```r
permTestTxIA_customPick(RS1 = NULL, txdb = NULL, type = 'mature',
                         customPick_function = NULL, ntimes = 50,
                         ev_function_1 = overlapCountsTxIA, ev_function_2 = overlapCountsTx, pval_z = FALSE, ...)
```

### Arguments

- **RS1**: The region set to be randomized. It should be in the GRanges or GRangesList format.
- **txdb**: A TxDb object.
- **type**: A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.
- **customPick_function**: A custom function needs to be inputted by users. The customPick function should have two arguments: a TxDb object and a character object of transcript ids. It returns a specified region over each transcript.
permTestTxIA_customPick

ntimes Randomization times.
ev_function_1 Evaluation function defines what statistic to be tested between RS1 and RS2. Default is overlapCountsTxIA.
ev_function_2 Evaluation function defines what statistic to be tested between each element in RSL and RS2. Default is overlapCountsTx.
pval_z Boolean. Default is FALSE. If FALSE, the p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, the p-value is calculated based on a z-test.
... Any additional parameters needed.

Details

permTestTxIA_customPick will assess the test statistic between RS1 and each region in RSL, and the relation between RS1 and RS2. Each RNA feature is only mapped with a part of region on its transcript (picked by the customPick_function). The output orig.ev is the weighted counts between RS1 and RS2. Each feature in RS1 related to n1 isoforms in RS2 and overlapped with n2 RS2 regions will contribute a value of n2/n1 to the total number of overlaps. This test function also randomizes input features per transcript. The set of randomized results is outputted as RSL. The overlapping counts between each set in RSL with RS2 is outputted as rand.ev.

Value

A list object, which is defined to be permTestTx_results class. It contains the following information:

- RSL: Randomized region sets of RS1.
- RS1: The feature set to be randomized.
- RS2: The region set to be compared with the feature set.
- orig.ev: The value of overlapping counts between RS1 and RS2.
- rand.ev: The values of overlapping counts between each element in RSL and RS2.
- pval: p-value of the test.
- zscore: Standard score of the test.

See Also

plotPermResults

Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
RS1 <- m6A_sites_data[1:500]

permTestTx_results <- permTestTxIA_customPick(RS1 = RS1,
txdb = txdb,
permTestTx_customAll

Perform permutation test

description

Perform permutation test for evaluating spatial association between region sets. This permutation test function receives two region sets and a set of randomized region sets of one of them. It evaluates if there is an association between these two region sets.

Usage

permTestTx_customAll(RSL = NULL, RS1 = NULL, RS2 = NULL, ev_function_1 = overlapCountsTx, ev_function_2 = overlapCountsTx, pval_z = FALSE, ...)

Arguments

RSL Randomized region sets of RS1. It should be a list object and each element should be in the GRanges or GRangesList format.
RS1 The region set. It should be in the GRanges or GRangesList format.
RS2 The region set to be compared with. It should be in the GRanges or GRangesList format.
ev_function_1 Evaluation function defines what statistic to be tested between RS1 and RS2. Default is overlapCountsTx.
ev_function_2 Evaluation function defines what statistic to be tested between each element in RSL and RS2. Default is overlapCountsTx.
pval_z Boolean. Default is FALSE. If FALSE, p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, p-value is calculated based on a z-test.
... Any additional parameters needed.

details

permTestTx_customAll will use evaluation function ev_function_1 to calculate the test statistic between RS1 and RS2, and use ev_function_2 to evaluate the statistic between RSL and RS2. It will also return a p-value and a z-score.

type = 'mature',
customPick_function = getStopCodon,
times = 5)
permTestTx_customPick

Value
A list object, which is defined to be permTestTx.results class. It contains the following items:

- **RSL**: Randomized region sets of RS1.
- **RS1**: The feature set to be randomized.
- **RS2**: The region set to be compared with the feature set.
- **orig.ev**: The value of overlapping counts between RS1 and RS2.
- **rand.ev**: The values of overlapping counts between each element in RSL and RS2.
- **pval**: p-value of the test.
- **zscore**: Standard score of the test.

Examples

```r
library(TxDB.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids1<- c("170")
RS1 <- randomizeTx(txdb = txdb, trans_ids = trans.ids1, random_num = 20, random_length = 100)
RS2 <- randomizeTx(txdb = txdb, trans_ids = trans.ids1, random_num = 20, random_length = 100)
trans.ids2 <- c("170", "782", "974", "1364", "1387")
RSL <- randomizeTx(txdb = txdb, trans_ids = trans.ids2, random_num = 20, random_length = 100, N = 10)
permTestTx_results <- permTestTx_customAll(RSL = RSL, RS1 = RS1, RS2 = RS2)
```

permTestTx_customPick  Perform permutation test

Description
Perform permutation test for evaluating spatial association between a feature set and the customPick regions. The latter is defined by the customPick_function argument provided by users.

Usage

```r
permTestTx_customPick(RS1 = NULL, txdb = NULL, type = "mature", customPick_function = NULL, ntimes = 50, ev_function_1 = overlapCountsTx, ev_function_2 = overlapCountsTx, pval_z = FALSE, ...)
```

Arguments

- **RS1**: The feature set to be randomized. It should be in the GRanges or GRangesList format.
- **txdb**: A TxDb object.
type
A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.

customPick_function
A custom function needs to be inputted by users. The custom function should have two arguments: a TxDb object and a character object of transcript ids. It returns a part of region of each transcript.

ntimes
Randomization times.

ev_function_1
Evaluation function defines what statistic to be tested between RS1 and RS2. Default is overlapCountsTx.

ev_function_2
Evaluation function defines what statistic to be tested between each element in RSL and RS2. Default is overlapCountsTx.

pval_z
Boolean. Default is FALSE. If FALSE, the p-value is calculated based on the number of random evaluations is larger or less than the initial evaluation. If TRUE, the p-value is calculated based on a z-test.

Details
Each feature in RS1 is only mapped with the customPick regions over its transcript (picked by the customPick_function). The output orig.ev is the number of features that have overlap with its customPick region. The set of randomized region sets is outputted as RSL. The overlapping counts between each set in RSL with RS2 is outputted as rand.ev.

Value
A list object, which is defined to be permTestTx.results class. It contains the following items:

- RSL: Randomized region sets of RS1.
- RS1: The feature set to be randomized.
- RS2: The region set to be compared with the feature set.
- orig.ev: The value of overlapping counts between RS1 and RS2.
- rand.ev: The values of overlapping counts between each element in RSL and RS2.
- pval: p-value of the test.
- zscore: Standard score of the test.

See Also
plotPermResults

Examples
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
exons.tx0 <- exonsBy(txdb)
trans.ids <- sample(names(exons.tx0), 100)
RS1 <- randomizeTx(txdb, trans.ids, random_num = 100, random_length = 200, type = 'CDS')
getCDS = function(txdb, trans.id){
cds.tx0 <- cdsBy(txdb, use.names=FALSE)
cds.names <- as.character(intersect(names(cds.tx0), trans.id))
cds = cds.tx0[cds.names]
return(cds)
}
permTestTx_results <- permTestTx_customPick(RS1,txdb, customPick_function = getCDS, ntimes = 5)

plotPermResults(permTestTx_results = NULL, breaks = 15, alpha = 0.05, test_type = "one-sided", binwidth = NULL)

Arguments

permTestTx_results
A permTestTx.results list object, which can be generated by the permTestTx function.

breaks
Histogram breaks. Default is 15.

alpha
Significance level.

test_type
The type of the test. This argument only receives either two options "one-sided" or "two-sided". Default is "one-sided".

binwidth
Histogram binwidth.

Value
A plot object.

See Also
permTestTx

Examples
file <- system.file(package="RgnTX", "extdata", "permTestTx_results.rds")
permTestTx_results <- readRDS(file)
p_a <- plotPermResults(permTestTx_results, binwidth = 1)
p_a

plotPermResults

Plot permutation test results

Description
Show a graphical representation of permutation test.

Usage

plotPermResults(permTestTx_results = NULL, breaks = 15, alpha = 0.05, test_type = "one-sided", binwidth = NULL)
plotShiftedZScoreTx | Plot shifted z scores

Description
Plot shifted z scores for permutation test results.

Usage
plotShiftedZScoreTx(shiftedZScoresTx_results)

Arguments
shiftedZScoresTx_results
A shiftedZScoreTx.results object.

Value
A plot.

See Also
shiftedZScoreTx

Examples
file <- system.file(package="RgnTX", "extdata", "shiftedZScoreTx_results1.rds")
shiftedZScoreTx_results <- readRDS(file)
p1 <- plotShiftedZScoreTx(shiftedZScoreTx_results)
p1

randomizeFeaturesTx | Randomize features into transcriptome

Description
Randomize features into transcriptome.

Usage
randomizeFeaturesTx(RS, txdb, type = "mature", N = 1, ...)
**Arguments**

- **RS**
  The feature to be randomized. It should be a GRanges or GRangesList object.

- **txdb**
  A TxDb object.

- **type**
  A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.

- **N**
  The number of iterations.

- **...**
  Any additional parameters needed.

**Value**

A GRangesList object. The name of each element is the id of the transcript where the corresponding range is located.

**See Also**

randomizeTransByOrder, randomizeFeaturesTxIA, randomizeTx

**Examples**

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
RS1 <- randomizeTx(txdb, trans.ids, random_num = 100, random_length = 100)
RS <- randomizeFeaturesTx(RS1, txdb, N = 1)
```

**randomizeFeaturesTxIA**  Randomize features into transcriptome

**Description**

Randomize features into transcriptome, especially for the features that have isoform ambiguity.

**Usage**

`randomizeFeaturesTxIA(RS, txdb, type = "mature", N = 1, ...)`

**Arguments**

- **RS**
  The feature being randomized. It should be a GRanges or GRangesList object.

- **txdb**
  A TxDb object.

- **type**
  A character object. Default is "mature". It accepts options "mature", "full", "fiveUTR", "CDS" or "threeUTR", with which one can get corresponding types of transcriptome regions.

- **N**
  Randomization times.

- **...**
  Any additional parameters needed.
randomizeTransByOrder

Value

A GRangesList object. The name of each element is the id of the transcript where the corresponding range is located.

See Also

randomizeTransByOrder, randomizeFeaturesTx, randomizeTx

Examples

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
RS1 <- m6A_sites_data[1:100]
RS <- randomizeFeaturesTxIA(RS1, txdb, N = 1)
```

Description

This function receives a GRangesList object and picks a random region within each list element of this object. The length of the region to be picked is decided by the input `random_length` argument.

Usage

```r
randomizeTransByOrder(regions_A, random_length = 20)
```

Arguments

- `regions_A`: A GRangesList object. The name of each list element should be the corresponding transcript id.
- `random_length`: A numeric object.

Value

A GRangesList object. The name of each list element should be the corresponding transcript id.

See Also

randomizeTx, randomizeFeaturesTx, randomizeFeaturesTxIA
randomizeTx

Examples

    library(TxDb.Hsapiens.UCSC.hg19.knownGene)
    txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
    exons.tx0 <- exonsBy(txdb)
    trans.ids <- sample(names(exons.tx0), 500)
    regions.A <- exons.tx0[trans.ids]
    RS <- randomizeTransByOrder(regions.A, random_length = 20)

randomizeTx

Get randomized regions over transcriptome

Description

Pick random regions over specified transcripts.

Usage

randomizeTx(txdb, trans_ids = 'all',
random_num = 100, random_length = 20, type = 'mature', N = 1, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txdb</td>
<td>A TxDb object.</td>
</tr>
<tr>
<td>trans_ids</td>
<td>The ids of transcripts, which should be a character object. Random regions will be picked from these transcripts. If this argument takes the default value 'all', the scope of picking random regions will be the whole transcriptome.</td>
</tr>
<tr>
<td>random_num</td>
<td>The number of regions to be picked.</td>
</tr>
<tr>
<td>random_length</td>
<td>The length of regions to be picked.</td>
</tr>
<tr>
<td>type</td>
<td>A character object. Default is &quot;mature&quot;. It accepts options &quot;mature&quot;, &quot;full&quot;, &quot;fiveUTR&quot;, &quot;CDS&quot; or &quot;threeUTR&quot;, with which one can get corresponding types of transcriptome regions.</td>
</tr>
<tr>
<td>N</td>
<td>Randomization times.</td>
</tr>
<tr>
<td>...</td>
<td>Any additional parameters needed.</td>
</tr>
</tbody>
</table>

Value

A GRangesList object. The name of each element is the id of the transcript where the corresponding range is located.

See Also

randomizeTransByOrder, randomizeFeaturesTx, randomizeFeaturesTxIA
Examples

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
trans.ids <- c("170", "782", "974", "1364", "1387")
RS1 <- randomizeTx(txdb, trans.ids, random_num = 100, random_length = 100)

shiftedZScoreTx  Calculate shifted z scores

Description

Calculate shifted z scores for permutation test results.

Usage

shiftedZScoreTx(permTestTx_results = NULL, txdb = NULL,
window = 200, step = 20, ev_function_1 = overlapCountsTx, ...)

Arguments

permTestTx_results  A permTestTx.results object.
txdb  A TxDb object.
window  The window of the whole shifting.
step  The step of each shifting.
ev_function_1  Evaluation function. Default is overlapCountsTx.
...  Any additional parameters needed.

Details

see examples in plotShiftedZScoreTx

Value

A list object, which is defined to be shiftedZScoreX.results class. It contains the following items:

- shifted.z.scores: Standard z-scores after shifting.
- window: Window of the whole shifting.
- step: Step of each shifting.
- original.z.score: Original standard score.

See Also

plotShiftedZScoreTx
Examples

```r
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
file <- system.file(package="RgnTX", "extdata/m6A_sites_data.rds")
m6A_sites_data <- readRDS(file)
RS1 <- m6A_sites_data[1:500]
permTestTx_results <- permTestTxIA_customPick(RS1 = RS1,
                                          txdb = txdb,
                                          customPick_function = getStopCodon,
                                          ntimes = 5)
shiftedZScoreTx_results <- shiftedZScoreTx(permTestTx_results, txdb = txdb,
                                          window = 2000,
                                          step = 200,
                                          ev_function_1 = overlapCountsTxIA)
```

shiftTx

**Shift over transcripts**

Description

Calculate positional shifting over transcript regions. This function accepts a feature set and outputs a region set from it. Each output region is from each input feature.

Usage

```r
shiftTx(regions, start, width, direction, strand)
```

Arguments

- **regions**: A feature set following the format indicated in vignette section 3. Either to be GRanges or GRangesList.
- **start**: Starting positions. Each value represents a starting position in each input feature.
- **width**: Widths. Each value represents a width of each region to be picked from each feature.
- **direction**: Either to be character "left" or "right", which means the direction to which the starting position is shifting. The former means moving to the direction of 5’ while the latter means moving to 3’.
- **strand**: The strand type of the transcripts. It receives "+" or ".".

Value

A Granges object.
Examples

# Take five transcripts.
# Extract the last 200 nt regions from their CDS part.
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
trans.id.pstv <- c("170", "782", "974", "1364", "1387")
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

# download the CDS part of all transcriptome
cds.tx0 <- cdsBy(txdb, use.names = FALSE)

# pick the CDS part of these five transcripts
cds.p <- cds.tx0[trans.id.pstv]

width <- 200
start <- as.numeric(max(end(cds.p)))
R.cds.last200 <- shiftTx(cds.p, start = start, width = width, direction = 'left', strand = "+")

vector2GRangesList  vector2GRangesList

Description

Generate GRangesList object from vectors. The output region set follows the format required by the main permutation test functions.

Usage

vector2GRangesList(RefSeqID, targetName, strand, blockSizes, targetStart)

Arguments

RefSeqID  The name of each element.
targetName  The seqnames of each range.
strand  The strand of each range.
blockSizes  The width of each range.
targetStart  The start coordinate of each range.

Value

A GRangesList object.

See Also

GRanges2GRangesList
<table>
<thead>
<tr>
<th>Function</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculateShift</td>
<td>2, 5</td>
</tr>
<tr>
<td>distanceTx</td>
<td>4, 14</td>
</tr>
<tr>
<td>extractRegions</td>
<td>3, 5</td>
</tr>
<tr>
<td>getFormatCorrect</td>
<td>6</td>
</tr>
<tr>
<td>getPermSpaceByFeatures</td>
<td>6, 7, 8</td>
</tr>
<tr>
<td>getPermSpaceByTxID</td>
<td>7, 7, 8</td>
</tr>
<tr>
<td>getPermSpaceByType</td>
<td>7, 8</td>
</tr>
<tr>
<td>getPvalZscore</td>
<td>9</td>
</tr>
<tr>
<td>getStopCodon</td>
<td>9</td>
</tr>
<tr>
<td>getTransInfo</td>
<td>10</td>
</tr>
<tr>
<td>GRanges2GRangesList</td>
<td>11, 11, 12, 30</td>
</tr>
<tr>
<td>GRangesList2GRanges</td>
<td>12</td>
</tr>
<tr>
<td>overlapCountsTx</td>
<td>4, 12, 13, 14</td>
</tr>
<tr>
<td>overlapCountsTxIA</td>
<td>13</td>
</tr>
<tr>
<td>overlapWidthTx</td>
<td>4, 14, 14</td>
</tr>
<tr>
<td>permTestTx</td>
<td>15, 16, 23</td>
</tr>
<tr>
<td>permTestTx_customAll</td>
<td>20</td>
</tr>
<tr>
<td>permTestTx_customPick</td>
<td>18, 21</td>
</tr>
<tr>
<td>permTestTxIA</td>
<td>16</td>
</tr>
<tr>
<td>permTestTxIA_customPick</td>
<td>18</td>
</tr>
<tr>
<td>plotPermResults</td>
<td>16, 18, 19, 22, 23</td>
</tr>
<tr>
<td>plotShiftedZScoreTx</td>
<td>24, 28</td>
</tr>
<tr>
<td>randomizeFeaturesTx</td>
<td>24, 26, 27</td>
</tr>
<tr>
<td>randomizeFeaturesTxIA</td>
<td>25, 25, 26, 27</td>
</tr>
<tr>
<td>randomizeTransByOrder</td>
<td>25, 26, 26, 27</td>
</tr>
<tr>
<td>randomizeTx</td>
<td>25, 26, 27</td>
</tr>
<tr>
<td>shiftedZScoreTx</td>
<td>24, 28</td>
</tr>
<tr>
<td>shiftTx</td>
<td>29</td>
</tr>
<tr>
<td>vector2GRangesList</td>
<td>30</td>
</tr>
</tbody>
</table>