Package 'atena'

April 10, 2023

Type Package

Title Analysis of Transposable Elements

Version 1.4.1

Description Quantify expression of transposable elements (TEs) from RNA-seq data through different methods, including ERVmap, TEtranscripts and Telescope. A common interface is provided to use each of these methods, which consists of building a parameter object, calling the quantification function with this object and getting a SummarizedExperiment object as output container of the quantified expression profiles. The implementation allows one to quantify TEs and gene transcripts in an integrated manner.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.1), SummarizedExperiment

Imports methods, stats, Matrix, BiocGenerics, BiocParallel, S4Vectors, IRanges, GenomicRanges, GenomicAlignments, Rsamtools, GenomeInfoDb, SQUAREM, sparseMatrixStats, AnnotationHub, scales, matrixStats

Suggests covr, BiocStyle, knitr, rmarkdown, RUnit

biocViews Transcription, Transcriptomics, RNASeq, Sequencing, Preprocessing, Software, GeneExpression, Coverage, DifferentialExpression, FunctionalGenomics

VignetteBuilder knitr

URL https://github.com/functionalgenomics/atena

BugReports https://github.com/functionalgenomics/atena/issues

RoxygenNote 7.2.3

Collate 'AllGenerics.R' 'AllClasses.R' 'ERVmap.R' 'TEtranscripts.R' 'Telescope.R' 'annotations.R' 'atena.R' 'atenaMethod.R' 'overlappingModes.R' 'qtex.R' 'utils.R' 'zzz.R'

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

git_branch RELEASE_3_16

git_last_commit c41a0fb

git_last_commit_date 2023-02-03

Date/Publication 2023-04-10

Author Beatriz Calvo-Serra [aut, cre],
Robert Castelo [aut]

Maintainer Beatriz Calvo-Serra <beatriz.calvo@upf.edu>

R topics documented:

`atena-package` *atena: analysis of transposable elements in R and Bioconductor*

Description

The atena package provides a complete re-implementation in R of three existing methods for the quantification of transposable element (TE) expression in order to facilitate its integration into Bioconductor workflows for the analysis of RNA-seq data. The three methods are TEtranscripts (Jin et al. (2015)), ERVmap (Tokuyama et al. (2018)) and Telescope (Bendall et al.(2019)).

Details

The main functions are:

- `TEtranscriptsParam` - build parameter objects of the class `TEtranscriptsParam-class` for the TEtranscripts expression quantification method
- `ERVmapParam` - build parameter objects of the class `ERVmapParam-class` for the ERVmap expression quantification method
- `TelescopeParam` - build parameter objects of the class `TelescopeParam-class` for the Telescope expression quantification method
- `qtex` - call the TE expression quantification method using a previously built parameter object

For detailed information on usage, see the package vignette, by typing `vignette("atena")`.

All questions and bug reports should be posted to the Bioconductor Support Site:

https://support.bioconductor.org

The code of the development version of the package is available at the GitHub repository:

https://github.com/functionalgenomics/atena

References

Jin Y et al. TEtranscripts: a package for including transposable elements in differential expression analysis of RNA-seq datasets. Bioinformatics. 2015;31(22):3593-3599. DOI: https://doi.org/10.1093/bioinformatics/btv422

Tokuyama M et al. ERVmap analysis reveals genome-wide transcription of human endogenous retroviruses. PNAS. 2018;115(50):12565-12572. DOI: https://doi.org/10.1073/pnas.1814589115

Bendall et al. Telescope: characterization of the retrotranscriptome by accurate estimation of transposable element expression. PLOS Comp. Biol. 2019;15(9):e1006453. DOI: https://doi.org/10.1371/journal.pcbi.1006453

`annotaTEs` *Get RepeatMasker UCSC annotations*

Description

The `annotaTEs()` function fetches RepeatMasker UCSC transposable element (TE) annotations using AnnotationHub and parses them.

Usage

```
annotaTEs(genome = "hg38", parsefun = rmskidentity)
```

Arguments

`genome`	The genome version of the desired RepeatMasker annotations (e.g. "hg38").
`parsefun`	A `function` to parse the annotations: • Function `rmskidentity` returns RepeatMasker annotations as present in AnnotationHub, without processing them. • Function `rmskbasicparser` parses annotations by removing low complexity regions, simple repeats, satellites, rRNA, scRNA, snRNA, srpRNA and tRNA. Also removes TEs with a strand different than "+" or "-". Modifies "repFamily" and "repClass" columns when a "?" is present or when they are defined as "Unknown" or "Other". Finally, assigns a unique id to each TE instance by adding the suffix "_dup" plus a number at the end of the "repName". • User-defined function. Input and output should be `GRanges` objects.

Details

Given a specific genome version, the `annotaTEs()` function fetches RepeatMasker annotations from UCSC Genome Browser using the AnnotationHub package. Since RepeatMasker not only provides TE annotations but also low complexity DNA sequences and other types of repeats, a specific `parsefun` can be set to parse these annotations (e.g. `rmskbasicparser` or a user-defined function). If no parsing is required, `parsefun` can be set to `rmskidentity`.

Value

A GRanges object with transposable element annotations.

See Also

`AnnotationHub`

Examples

```
rmsk_gr <- annotaTEs(genome = "hg38", parsefun = rmskidentity)
```

`atenaParam-class` *atena parameter class*

Description

This is a class for storing parameters to quantify TE (and gene) expression using the atena method. It is a subclass of the 'QuantifyParam-class'.

Build an object of the class `atenaParam`.

Usage

```
atenaParam(
  bfl,
  teFeatures,
  aggregateby = character(0),
  ovMode = "ovUnion",
  geneFeatures = NA,
  singleEnd = TRUE,
  strandMode = 1L,
  ignoreStrand = FALSE,
  fragments = TRUE,
  pi_prior = 0L,
  theta_prior = 0L,
  em_epsilon = 1e-07,
  maxIter = 100L,
  reassign_mode = "exclude",
```

```
  conf_prob = 0.9
)

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

Arguments

bfl	A BamFile or BamFileList object, or a character string vector of BAM filenames.
teFeatures	A GRanges or GRangesList object. Elements in this object should have names, which are used as a grouping factor for genomic ranges forming a common locus. This grouping is performed previous to TE expression quantification, unlike the aggregation of quantifications performed when the aggregateby parameter is specified, which is performed after individual TE instances are quantified.
aggregateby	Character vector with column names from the annotation to be used to aggregate quantifications. By default, this is an empty vector, which means that the names of the input GRanges or GRangesList object given in the teFeatures parameter are used to aggregate quantifications.
ovMode	Character vector indicating the overlapping mode. Available options are: "ovUnion" (default) and "ovIntersectionStrict", which implement the corresponding methods from HTSeq (https://htseq.readthedocs.io/en/release_0.11.1/count.html). Ambiguous alignments (alignments overlapping > 1 feature) are not counted.
geneFeatures	A GRanges or GRangesList object with the gene annotated features to be quantified. Unique reads are first tallied with respect to these gene features whereas multi-mapping reads are preferentially assigned to TEs. Elements should have names indicating the gene name/id. In case that geneFeatures contains a metadata column named type, only the elements with type = exon are considered for the analysis. Then, exon counts are summarized to the gene level.
singleEnd	(Default TRUE) Logical value indicating if reads are single (TRUE) or paired-end (FALSE).
strandMode	(Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on GAlignmentPairs objects that controls the behavior of the strand getter. See GAlignmentPairs class for further detail. If singleEnd = TRUE, then strandMode is ignored.
ignoreStrand	(Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When ignoreStrand = FALSE, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when ignoreStrand = TRUE the strand is not considered.
fragments	(Default TRUE) A logical; applied to paired-end data only. When fragments=FALSE, the read-counting method only counts 'mated pairs' from opposite strands (non-ambiguous properly paired reads), while when fragments=TRUE same-strand

pairs, singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are also counted (see "Pairing criteria" in `readGAlignments()`). For further details see `summarizeOverlaps()`.

`pi_prior` (Default 0) A positive numeric object indicating the prior on pi. The same prior can be specified for all features setting `pi_prior` as a scalar, or each feature can have a specific prior by setting `pi_prior` as a vector with `names()` corresponding to all feature names. Setting a pi prior is equivalent to adding n unique reads.

`theta_prior` (Default 0) A positive numeric object indicating the prior on Q. The same prior can be specified for all features setting `theta_prior` as a scalar, or each feature can have a specific prior by setting `theta_prior` as a vector with `names()` corresponding to all feature names. Equivalent to adding n non-unique reads.

`em_epsilon` (Default 1e-7) A numeric scalar indicating the EM Algorithm Epsilon cutoff.

`maxIter` A positive integer scalar storing the maximum number of iterations of the EM SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the `maxiter` parameter of the `squarem()` function.

`reassign_mode` (Default 'exclude') Character vector indicating reassignment mode after EM step. Available methods are 'exclude' (reads with more than one best assignment are excluded from the final counts), 'choose' (when reads have more than one best assignment, one of them is randomly chosen), 'average' (the read count is divided evenly among the best assignments) and 'conf' (only assignments that exceed a certain threshold -defined by `conf_prob` parameter- are accepted, then the read count is proportionally divided among the assignments above `conf_prob`).

`conf_prob` (Default 0.9) Minimum probability for high confidence assignment.

`object` A atenaParam object.

Details

This is the constructor function for objects of the class `atenaParam-class`. This type of object is the input to the function `qtex()` for quantifying expression of transposable elements, which will call the atena method with this type of object. The atena method uses a multiple '__no_feature' approach in which as many '__no_feature' features as different overlapping patterns of multimapping reads in the overlapping matrix are used to represent alignments mapping outside annotations.

Value

A atenaParam object.

Slots

`singleEnd` (Default TRUE) Logical value indicating if reads are single (`TRUE`) or paired-end (`FALSE`).

`strandMode` (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on `GAlignmentPairs` objects that controls the behavior of the strand getter. See `GAlignmentPairs` class for further detail. If `singleEnd = TRUE`, then `strandMode` is ignored.

`ignoreStrand` (Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When `ignoreStrand = FALSE`, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when `ignoreStrand = TRUE` the strand is not considered.

`fragments` (Default TRUE) A logical; applied to paired-end data only. When `fragments=FALSE`, the read-counting method only counts 'mated pairs' from opposite strands (non-ambiguous properly paired reads), while when `fragments=TRUE` same-strand pairs, singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are also counted (see "Pairing criteria" in `readGAlignments()`). For further details see `summarizeOverlaps()`.

`pi_prior` (Default 0) A positive numeric object indicating the prior on pi. The same prior can be specified for all features setting `pi_prior` as a scalar, or each feature can have a specific prior by setting `pi_prior` as a vector with `names()` corresponding to all feature names. Setting a pi prior is equivalent to adding n unique reads.

`theta_prior` (Default 0) A positive numeric object indicating the prior on Q. The same prior can be specified for all features setting `theta_prior` as a scalar, or each feature can have a specific prior by setting `theta_prior` as a vector with `names()` corresponding to all feature names. Equivalent to adding n non-unique reads.

`em_epsilon` (Default 1e-7) A numeric scalar indicating the EM Algorithm Epsilon cutoff.

`maxIter` A positive integer scalar storing the maximum number of iterations of the EM SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the `maxiter` parameter of the `squarem()` function.

`reassign_mode` (Default 'exclude') Character vector indicating reassignment mode after EM step. Available methods are 'exclude' (reads with more than one best assignment are excluded from the final counts), 'choose' (when reads have more than one best assignment, one of them is randomly chosen), 'average' (the read count is divided evenly among the best assignments) and 'conf' (only assignments that exceed a certain threshold -defined by `conf_prob` parameter- are accepted, then the read count is proportionally divided among the assignments above `conf_prob`).

`conf_prob` (Default 0.9) Minimum probability for high confidence assignment.

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
gene_annot <- readRDS(file = system.file("extdata", "Top50genes.rds",
                                         package="atena"))
atpar <- atenaParam(bfl=bamfiles, teFeatures=TE_annot,
                        geneFeatures = gene_annot,
                        singleEnd = TRUE, ignoreStrand=TRUE)
atpar
```

ERVmapParam-class *ERVmap parameter class*

Description

This is a class for storing parameters provided to the ERVmap algorithm. It is a subclass of the 'QuantifyParam-class'.

Build an object of the class ERVmapParam

Usage

```
ERVmapParam(
  bfl,
  teFeatures,
  aggregateby = character(0),
  ovMode = "ovUnion",
  geneFeatures = NA,
  singleEnd = TRUE,
  ignoreStrand = TRUE,
  strandMode = 1L,
  fragments = !singleEnd,
  maxMismatchRate = 0.02,
  suboptimalAlignmentTag = "auto",
  suboptimalAlignmentCutoff = 5,
  geneCountMode = "all"
)

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

Arguments

bfl	A BamFile or BamFileList object, or a character string vector of BAM filenames.
teFeatures	A GRanges or GRangesList object with the transposable element (TE) annotated features to be quantified. Elements in this object should have names, which are used as a grouping factor for genomic ranges forming a common locus, unless other metadata column names are specified in the aggregateby parameter.
aggregateby	Character vector with column names in the annotation to be used to aggregate quantifications. By default, this is an empty vector, which means that the names of the input GRanges or GRangesList object given in the teFeatures parameter are used to aggregate quantifications.
ovMode	Character vector indicating the overlapping mode. Available options are: "ovUnion" (default) and "ovIntersectionStrict", which implement the corresponding methods from HTSeq (https://htseq.readthedocs.io/en/release_0.11.

1/count.html). Ambiguous alignments (alignments overlapping > 1 feature) are addressed as in the original ERVmap algorithm.

geneFeatures
: A GRanges or GRangesList object with the gene annotated features to be quantified. Overlaps with unique reads are first tallied with respect to these gene features. Elements should have names indicating the gene name/id. In case that geneFeatures contains a metadata column named type, only the elements with type = exon are considered for the analysis. Then, exon counts are summarized to the gene level.

singleEnd
: (Default TRUE) Logical value indicating if reads are single (TRUE) or paired-end (FALSE).

ignoreStrand
: (Default TRUE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When ignore_strand = FALSE, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when ignoreStrand = TRUE the strand is not considered.

strandMode
: (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on GAlignmentPairs objects that controls the behavior of the strand getter. See GAlignmentPairs class for further detail. If singleEnd = TRUE, then strandMode is ignored.

fragments
: (Default not singleEnd) A logical; applied to paired-end data only. When fragments=TRUE, the read-counting method in the original ERVmap algorithm is applied: each mate of a paired-end read is counted (including ambiguous and not properly paired reads). When fragments=FALSE, if the two mates of a paired-end read map to the same element, they are counted as a single hit and singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are not counted (see "Pairing criteria" in readGAlignments()).

maxMismatchRate
: (Default 0.02) Numeric value storing the maximum mismatch rate employed by the ERVmap algorithm to discard aligned reads whose rate of sum of hard and soft clipping or whose rate of the edit distance over the genome reference to the length of the read is above this threshold.

suboptimalAlignmentTag
: (Default "auto") Character string storing the tag name in the BAM files that stores the suboptimal alignment score used in the third filter of ERVmap; see Tokuyama et al. (2018). The default, suboptimalAlignmentTag="auto", first extracts the name of the read mapper software from one or more BAM files. If BAM files were generated by BWA, the suboptimal alignment scores are obtained from a tag called XS. For other read mappers, the suboptimal alignment score is considered to be missing since, except from BWA, no other aligner provides a tag with suboptimal alignment scores. In this case, the available secondary alignments are used to implement an analogous approach to that of the third ERVmap filter. When suboptimalAlignmentTag="none", it also performs the latter approach even when the tag XS is available. When this parameter is different from "auto" and "none", a tag with the given name is used to extract the suboptimal alignment score.

`suboptimalAlignmentCutoff`
: (Default 5) Numeric value storing the cutoff above which the difference between the alignment score and the suboptimal alignment score is considered sufficiently large to retain the alignment. When this value is set to `NA`, the filtering step based on suboptimal alignment scores is skipped.

`geneCountMode`
: (Default `"all"`) Character string indicating if the ERVmap read filters applied to quantify TEs expression should also be applied when quantifying gene expression (`"ervmap"`) or not (`"all"`), in which case all primary alignments mapping to genes are counted.

`object`
: A ERVmapParam object.

Details

This is the constructor function for objects of the class `ERVmapParam-class`. This type of object is the input to the function `qtex()` for quantifying expression of transposable elements using the ERVmap method Tokuyama et al. (2018). The ERVmap algorithm processes reads following conservative filtering criteria to provide reliable raw count data for each TE.

Value

A ERVmapParam object.

Slots

`readMapper` The name of the software used to align reads, obtained from the BAM file header.

`singleEnd` (Default FALSE) Logical value indicating if reads are single (`TRUE`) or paired-end (`FALSE`).

`strandMode` (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on `GAlignmentPairs` objects that controls the behavior of the strand getter. See `GAlignmentPairs` class for further detail. If `singleEnd = TRUE`, then `strandMode` #' is ignored.

`ignoreStrand` (Default TRUE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and TEs in the annotations. When `ignore_strand = FALSE`, only those reads which overlap the TE and are on the same strand are counted. On the contrary, when `ignore_strand = TRUE`, any read overlapping an element in `teFeatures` is counted regardless of the strand.

`fragments` (Default not `singleEnd`) A logical; applied to paired-end data only. When `fragments=TRUE`, the read-counting method in the original ERVmap algorithm is applied: each mate of a paired-end read is counted (including ambiguous and not properly paired reads). When `fragments=FALSE`, if the two mates of a paired-end read map to the same element, they are counted as a single hit and singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are not counted (see "Pairing criteria" in `readGAlignments()`).

`maxMismatchRate` (Default 0.02) Numeric value storing the maximum mismatch rate employed by the ERVmap algorithm to discard aligned reads whose rate of sum of hard and soft clipping, or of the edit distance over the genome reference, to the length of the read is above this threshold.

suboptimalAlignmentTag (Default "auto") Character string storing the tag name in the BAM files that stores the suboptimal alignment score used in the third filter of ERVmap; see Tokuyama et al. (2018). The default, suboptimalAlignmentTag="auto", assumes that either the BAM files were generated by BWA and include a tag called XS that stores the suboptimal alignment score or, if the XS tag is not available, then it uses the available secondary alignments to implement an analogous approach to that of the third ERVmap filter. When suboptimalAlignmentTag="none", it also performs the latter approach even when the tag XS is available. When this parameter is different from "auto" and "none", a tag with the given name is used to extract the suboptimal alignment score. The absence of that tag will prompt an error.

suboptimalAlignmentCutoff (Default 5) Numeric value storing the cutoff above which the difference between the alignment score and the suboptimal alignment score is considered sufficiently large to retain the alignment. When this value is set to NA, then the filtering step based on suboptimal alignment scores is skipped.

geneCountMode (Default "all") Character string indicating if the ERVmap read filters applied to quantify TEs expression should also be applied when quantifying gene expression ("ervmap") or not ("all"), in which case all primary alignments mapping to genes are counted.

References

Tokuyama M et al. ERVmap analysis reveals genome-wide transcription of human endogenous retroviruses. PNAS. 2018;115(50):12565-12572. DOI: https://doi.org/10.1073/pnas.1814589115

Tokuyama M et al. ERVmap analysis reveals genome-wide transcription of human endogenous retroviruses. PNAS. 2018;115(50):12565-12572. DOI: https://doi.org/10.1073/pnas.1814589115

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
empar <- ERVmapParam(bamfiles, teFeatures = TE_annot, singleEnd = TRUE,
                     ignoreStrand = TRUE, suboptimalAlignmentCutoff=NA)
empar
```

---

ovUnion *Pre-defined overlapping mode functions*

Description

The following functions control the way in which overlaps between aligned reads and annotated features are resolved when an aligned read overlaps more than one feature on the same locus:

Usage

```
ovUnion(reads, features, ignoreStrand, inter.feature = TRUE)

ovIntersectionStrict(reads, features, ignoreStrand, inter.feature = TRUE)
```

Arguments

reads	A GAlignments, GAlignmentList or a GAlignmentPairs object.
features	A GRanges object with annotated features.
ignoreStrand	(Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When ignoreStrand = FALSE, an aligned read will be considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic ranges on the same strand, while when ignoreStrand = TRUE the strand will not be considered.
inter.feature	When TRUE, ambiguous alignments (alignments overlapping > 1 features) are removed and not counted. When inter.feature is set to FALSE, these ambiguous overlaps are taken into account and addressed differently depending on the TE quantification.

Details

- ovUnion(): (default)
- ovIntersectionStrict():
- User supplied: a function taking the same parameters as the previous three functions and returning a Hits object.

They take the following parameters:

These functions are given to the mode parameter of the qtex() function and are similar to the functions Union() and IntersectionStrict() from the GenomicAlignments package, with the difference that instead of returning counts of reads overlapping annotated features, they return the actual overlaps, because the counting is deferred to other algorithms that follow some specific strategy when a read maps to more than one feature. For this same reason, these functions lack the inter.feature argument found in the corresponding functions from the GenomicAlignments package.

Value

A Hits object; see the Hits-class manual page.

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
tspar <- TelescopeParam(bfl=bamfiles, teFeatures=TE_annot,
                        singleEnd = TRUE, ignoreStrand=TRUE,
                        minOverlFract=0L)
tsSE <- qtex(tspar, mode=ovIntersectionStrict)
```

qtex,ERVmapParam-method

Quantify transposable element expression

Description

The `qtex()` method quantifies transposable element expression.

Usage

```
## S4 method for signature 'ERVmapParam'
qtex(
  x,
  phenodata = NULL,
  mode = ovUnion,
  yieldSize = 1000000L,
  verbose = 1,
  BPPARAM = SerialParam(progressbar = ifelse(verbose == 1, TRUE, FALSE))
)

## S4 method for signature 'TEtranscriptsParam'
qtex(
  x,
  phenodata = NULL,
  mode = ovUnion,
  yieldSize = 1000000L,
  BPPARAM = SerialParam(progressbar = TRUE)
)

## S4 method for signature 'TelescopeParam'
qtex(
  x,
  phenodata = NULL,
  mode = ovUnion,
  yieldSize = 1000000L,
  BPPARAM = SerialParam(progressbar = TRUE)
)

## S4 method for signature 'atenaParam'
qtex(
  x,
  phenodata = NULL,
  mode = ovUnion,
  yieldSize = 1000000L,
  BPPARAM = SerialParam(progressbar = TRUE)
)
```

Arguments

x	An AtenaParam object of one of the following subclasses: • A ERVmapParam object built using the constructor function ERVmapParam(). This object will trigger qtex() to use the algorithm by Tokuyama et al. (2018). • A TelescopeParam object built using the constructor function TelescopeParam(). This object will trigger qtex() to use the algorithm by Bendall et al. (2019).
phenodata	A data.frame or DataFrame object storing phenotypic data to include in the resulting SummarizedExperiment object. If phenodata is set, its row names will become the column names of the resulting SummarizedExperiment object.
mode	One of the pre-defined overlapping methods such as ovUnion(), ovIntersectionStrict or a user-supplied overlapping function. For a user-supplied overlapping function, the input parameters must match those of the pre-defined methods and the function must return a Hits object with subject hits matching the annotated features. This parameter is analogous to the mode parameter of the summarizeOverlaps() function from the GenomicAlignments package.
yieldSize	Field inherited from BamFile. The method for signature ERVmapParam() reads the BAM file by chunks. yieldSize represents the number of records (chunk size) to yield each time the file is read.
verbose	(Default 1). When verbose > 1, detailed information on the quantification steps is provided. Warnings are always present regardless of the value of verbose.
BPPARAM	An object of a BiocParallelParam subclass to configure the parallel execution of the code. By default, a SerialParam object is used, which does not use any parallelization, with the flag progress=TRUE to show progress through the calculations.

Details

Giving some AtenaParam object sub-class as input, the qtex() method quantifies the expression of transposable elements (TEs). The particular algorithm to perform the quantification will be selected depending on the specific sub-class of input AtenaParam object, see argument x above.

Value

A SummarizedExperiment object.

References

Tokuyama M et al. ERVmap analysis reveals genome-wide transcription of human endogenous retroviruses. PNAS, 115(50):12565-12572, 2018. https://doi.org/10.1073/pnas.1814589115

Bendall ML et al. Telescope: characterization of the retrotranscriptome by accurate estimation of transposable element expression. PLOS Computational Biology, 15:e1006453, 2019. https://doi.org/10.1371/journal.pcbi.1006453

See Also

ERVmapParam TelescopeParam

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
gene_annot <- readRDS(file = system.file("extdata", "Top50genes.rds",
                                         package="atena"))
tspar <- TelescopeParam(bfl=bamfiles, teFeatures=TE_annot,
                        geneFeatures = gene_annot,
                        singleEnd = TRUE, ignoreStrand=TRUE)
tsSE <- qtex(tspar)
```

QuantifyParam-class *Atena parameter class*

Description

This is a virtual class from which other classes are derived for storing parameters provided to quantification methods of transposable elements from RNA-seq data.

Usage

```
## S4 method for signature 'QuantifyParam'
path(object)

## S4 method for signature 'QuantifyParam'
features(object)
```

Arguments

`object`	A QuantifyParam object.

Value

`path()`: Filesystem paths to the BAM files in the input parameter object.

`features()`: The `GenomicRanges` or `GenomicRangesList` object with the features in the input parameter object.

Slots

`bfl` A BamFileList object.

`features` A GRanges object.

`aggregateby` Character vector with column names in the annotation to be used to aggregate quantifications.

ovMode Character vector indicating the overlapping mode. Available options are: "ovUnion" (default) and "ovIntersectionStrict", which implement the corresponding methods from HTSeq (https://htseq.readthedocs.io/en/release_0.11.1/count.html). In the TEtranscripts, ERVmap and Telescope methods ambiguous alignments (alignments overlapping > 1 feature) are addressed differently depending on the method. In the atena method, those overlaps are not counted.

See Also

ERVmapParam-class TelescopeParam-class TEtranscriptsParam-class atenaParam-class

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
ttpar <- TEtranscriptsParam(bamfiles, teFeatures=TE_annot, singleEnd=TRUE,
                            ignoreStrand=TRUE, aggregateby = c("repName"))
path(ttpar)
```

---

rmskbasicparser *Parser of RepeatMasker annotations*

Description

Parser of RepeatMasker annotations

Usage

```
rmskbasicparser(gr)
```

Arguments

gr	A GRanges object with RepeatMasker annotations from AnnotationHub

Details

Parses annotations by removing low complexity regions, simple repeats, satellites, rRNA, scRNA, snRNA, srpRNA and tRNA. Also removes TEs with a strand different than "+" or "-". Modifies "repFamily" and "repClass" columns when a "?" is present or when they are defined as "Unknown" or "Other". Finally, assigns a unique id to each TE instance by adding the suffix "_dup" plus a number at the end of the "repName".

Value

A GRanges object.

Examples

```
rmsk_gr <- annotaTEs(genome = "dm6", parsefun = rmskbasicparser)
```

`rmskidentity` *Identity function for parsefun*

Description

Identity function for parsefun

Usage

```
rmskidentity(gr)
```

Arguments

`gr`	A GRanges object.

Details

Identity function: returns the GRanges object without any modification.

Value

A GRanges object.

Examples

```
rmsk_gr <- annotaTEs(genome = "dm6", parsefun = rmskidentity)
```

`TelescopeParam-class` *Telescope parameter class*

Description

This is a class for storing parameters provided to the Telescope algorithm.

Build an object of the class `TelescopeParam`.

Usage

```
TelescopeParam(
  bfl,
  teFeatures,
  aggregateby = character(0),
  ovMode = "ovUnion",
  geneFeatures = NA,
  singleEnd = TRUE,
  strandMode = 1L,
  ignoreStrand = FALSE,
  fragments = FALSE,
  minOverlFract = 0.2,
  pi_prior = 0L,
  theta_prior = 0L,
  em_epsilon = 1e-07,
  maxIter = 100L,
  reassign_mode = "exclude",
  conf_prob = 0.9
)

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

Arguments

bfl	A BamFile or BamFileList object, or a character string vector of BAM filenames.
teFeatures	A GRanges or GRangesList object. Elements in this object should have names, which are used as a grouping factor for genomic ranges forming a common locus (equivalent to "locus" column in Telescope). This grouping is performed previous to TE expression quantification, unlike the aggregation of quantifications performed when the aggregateby parameter is specified, which is performed after individual TE instances are quantified.
aggregateby	Character vector with column names from the annotation to be used to aggregate quantifications. By default, this is an empty vector, which means that the names of the input GRanges or GRangesList object given in the teFeatures parameter are used to aggregate quantifications.
ovMode	Character vector indicating the overlapping mode. Available options are: "ovUnion" (default) and "ovIntersectionStrict", which implement the corresponding methods from HTSeq (https://htseq.readthedocs.io/en/release_0.11.1/count.html). Ambiguous alignments (alignments overlapping > 1 feature) are addressed as in the original Telescope method: the overlap with the longest overlapping length is kept.
geneFeatures	A GRanges or GRangesList object with the gene annotated features to be quantified. The TEtranscripts approach for gene expression quantification is used, in which overlaps with multi-mapping reads are preferentially assigned to TEs. Elements should have names indicating the gene name/id. In case that geneFeatures

contains a metadata column named `type`, only the elements with `type = exon` are considered for the analysis. Then, exon counts are summarized to the gene level.

`singleEnd` (Default TRUE) Logical value indicating if reads are single (`TRUE`) or paired-end (`FALSE`).

`strandMode` (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on `GAlignmentPairs` objects that controls the behavior of the strand getter. See `GAlignmentPairs` class for further detail. If `singleEnd = TRUE`, then `strandMode` is ignored.

`ignoreStrand` (Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When `ignoreStrand = FALSE`, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when `ignoreStrand = TRUE` the strand is not considered.

`fragments` (Default FALSE) A logical; applied to paired-end data only. When `fragments=FALSE`, the read-counting method only counts 'mated pairs' from opposite strands (non-ambiguous properly paired reads), while when `fragments=TRUE` same-strand pairs, singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are also counted (see "Pairing criteria" in `readGAlignments()`). `fragments=TRUE` is equivalent to the original Telescope algorithm. For further details see `summarizeOverlaps()`.

`minOverlFract` (Default 0.2) A numeric scalar. `minOverlFract` is multiplied by the read length and the resulting value is used to discard alignments for which the overlapping length (number of base pairs the alignment and the feature overlap) is lower. When no minimum overlap is required, set `minOverlFract = 0`.

`pi_prior` (Default 0) A positive integer scalar indicating the prior on pi. This is equivalent to adding n unique reads.

`theta_prior` (Default 0) A positive integer scalar storing the prior on Q. Equivalent to adding n non-unique reads.

`em_epsilon` (Default 1e-7) A numeric scalar indicating the EM Algorithm Epsilon cutoff.

`maxIter` A positive integer scalar storing the maximum number of iterations of the EM SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the `maxiter` parameter of the `squarem()` function.

`reassign_mode` (Default 'exclude') Character vector indicating reassignment mode after EM step. Available methods are 'exclude' (reads with more than one best assignment are excluded from the final counts), 'choose' (when reads have more than one best assignment, one of them is randomly chosen), 'average' (the read count is divided evenly among the best assignments) and 'conf' (only assignments that exceed a certain threshold -defined by `conf_prob` parameter- are accepted, then the read count is proportionally divided among the assignments above `conf_prob`).

`conf_prob` (Default 0.9) Minimum probability for high confidence assignment.

`object` A TelescopeParam object.

Details

This is the constructor function for objects of the class `TelescopeParam-class`. This type of object is the input to the function `qtex()` for quantifying expression of transposable elements, which will call the Telescope algorithm Bendall et al. (2019) with this type of object.

Value

A TelescopeParam object.

Slots

`singleEnd` (Default TRUE) Logical value indicating if reads are single (`TRUE`) or paired-end (`FALSE`).

`strandMode` (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on `GAlignmentPairs` objects that controls the behavior of the strand getter. See `GAlignmentPairs` class for further detail. If `singleEnd = TRUE`, then `strandMode` is ignored.

`ignoreStrand` (Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When `ignoreStrand = FALSE`, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when `ignoreStrand = TRUE` the strand is not considered.

`fragments` (Default FALSE) A logical; applied to paired-end data only. When `fragments=FALSE`, the read-counting method only counts 'mated pairs' from opposite strands (non-ambiguous properly paired reads), while when `fragments=TRUE` same-strand pairs, singletons, reads with unmapped pairs and other ambiguous or not properly paired fragments are also counted (see "Pairing criteria" in `readGAlignments()`). `fragments=TRUE` is equivalent to the original Telescope algorithm. For further details see `summarizeOverlaps()`.

`minOverlFract` (Default 0.2) A numeric scalar. `minOverlFract` is multiplied by the read length and the resulting value is used to discard alignments for which the overlapping length (number of base pairs the alignment and the feature overlap) is lower. When no minimum overlap is required, set `minOverlFract = 0`.

`pi_prior` (Default 0) A positive integer scalar indicating the prior on pi. This is equivalent to adding n unique reads.

`theta_prior` (Default 0) A positive integer scalar storing the prior on Q. Equivalent to adding n non-unique reads.

`em_epsilon` (Default 1e-7) A numeric scalar indicating the EM Algorithm Epsilon cutoff.

`maxIter` A positive integer scalar storing the maximum number of iterations of the EM SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the `maxiter` parameter of the `squarem()` function.

`reassign_mode` (Default 'exclude') Character vector indicating reassignment mode after EM step. Available methods are 'exclude' (reads with more than one best assignment are excluded from the final counts), 'choose' (when reads have more than one best assignment, one of them is randomly chosen), 'average' (the read count is divided evenly among the best assignments) and 'conf' (only assignments that exceed a certain threshold -defined by `conf_prob` parameter- are accepted, then the read count is proportionally divided among the assignments above `conf_prob`).

`conf_prob` (Default 0.9) Minimum probability for high confidence assignment.

References

Bendall et al. Telescope: characterization of the retrotranscriptome by accurate estimation of transposable element expression. PLOS Comp. Biol. 2019;15(9):e1006453. DOI: https://doi.org/10.1371/journal.pcbi.1006453

Bendall et al. Telescope: characterization of the retrotranscriptome by accurate estimation of transposable element expression. PLOS Comp. Biol. 2019;15(9):e1006453. DOI: https://doi.org/10.1371/journal.pcbi.1006453

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
gene_annot <- readRDS(file = system.file("extdata", "Top50genes.rds",
                                         package="atena"))
tspar <- TelescopeParam(bfl=bamfiles, teFeatures=TE_annot,
                        geneFeatures = gene_annot,
                        singleEnd = TRUE, ignoreStrand=TRUE)
tspar
```

TEtranscriptsParam-class

TEtranscripts parameter class

Description

This is a class for storing parameters provided to the TEtranscripts algorithm. It is a subclass of the 'QuantifyParam-class'.

Build an object of the class `TEtranscriptsParam`

Usage

```
TEtranscriptsParam(
  bfl,
  teFeatures,
  aggregateby = character(0),
  ovMode = "ovUnion",
  geneFeatures = NA,
  singleEnd = TRUE,
  ignoreStrand = FALSE,
  strandMode = 1L,
  fragments = TRUE,
  tolerance = 1e-04,
  maxIter = 100L
)
```

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

Arguments

bfl	a character string vector of BAM file names.
teFeatures	A GRanges or GRangesList object with the TE annotated features to be quantified. Elements in this object should have names, which are used as a grouping factor for genomic ranges forming a common locus, unless other metadata column names are specified in the aggregateby parameter.
aggregateby	Character vector with column names from the annotation to be used to aggregate quantifications. By default, this is an empty vector, which means that the names of the input GRanges or GRangesList object given in the teFeatures parameter are used to aggregate quantifications.
ovMode	Character vector indicating the overlapping mode. Available options are: "ovUnion" (default) and "ovIntersectionStrict", which implement the corresponding methods from HTSeq (https://htseq.readthedocs.io/en/release_0.11.1/count.html). Ambiguous alignments (alignments overlapping > 1 feature) are addressed as in the original TEtranscripts method.
geneFeatures	A GRanges or GRangesList object with the gene annotated features to be quantified. Following the TEtranscripts algorithm, overlaps with unique reads are first tallied with respect to these gene features. Elements should have names indicating the gene name/id. In case that geneFeatures contains a metadata column named type, only the elements with type = exon are considered for the analysis. Then, exon counts are summarized to the gene level.
singleEnd	(Default TRUE) Logical value indicating if reads are single (TRUE) or paired-end (FALSE).
ignoreStrand	(Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When ignoreStrand = FALSE, an aligned read is considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic range on the same strand, while when ignoreStrand = TRUE the strand is not considered.
strandMode	(Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on GAlignmentPairs objects that controls the behavior of the strand getter. See GAlignmentPairs class for further detail. If singleEnd = TRUE, then strandMode is ignored.
fragments	(Default TRUE) A logical; applied to paired-end data only. In both cases (fragments=FALSE and fragments=TRUE), the read-counting method discards not properly paired reads. Moreover, when fragments=FALSE, only non-ambiguous properly paired reads are counted. When fragments=TRUE, ambiguous reads are also counted (see "Pairing criteria" in readGAlignments()). fragments=TRUE is equivalent to the behavior of the TEtranscripts algorithm. For further details see summarizeOverlaps().

tolerance	A positive numeric scalar storing the minimum tolerance above which the SQUAREM algorithm (Du and Varadhan, 2020) keeps iterating. Default is 1e-4 and this value is passed to the tol parameter of the squarem() function.
maxIter	A positive integer scalar storing the maximum number of iterations of the SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the maxiter parameter of the squarem() function.
object	A TEtranscriptsParam object.

Details

This is the constructor function for objects of the class TEtranscriptsParam-class. This type of object is the input to the function qtex() for quantifying expression of transposable elements using the TEtranscripts method Jin et al. (2015). The TEtranscripts algorithm quantifies TE expression by using an EM algorithm to optimally distribute ambiguously mapped reads.

Value

A TEtranscriptsParam object.

Slots

singleEnd (Default FALSE) Logical value indicating if reads are single (TRUE) or paired-end (FALSE).

ignoreStrand (Default FALSE) A logical which defines if the strand should be taken into consideration when computing the overlap between reads and annotated features. When ignoreStrand = FALSE, an aligned read will be considered to be overlapping an annotated feature as long as they have a non-empty intersecting genomic ranges on the same strand, while when ignoreStrand = TRUE the strand will not be considered.

strandMode (Default 1) Numeric vector which can take values 0, 1 or 2. The strand mode is a per-object switch on GAlignmentPairs objects that controls the behavior of the strand getter. See GAlignmentPairs class for further detail. If singleEnd = TRUE, then use either strandMode = NULL or do not specify the strandMode parameter.

fragments (Default TRUE) A logical; applied to paired-end data only. In both cases (fragments=FALSE and fragments=TRUE), the read-counting method discards not properly paired reads. Moreover, when fragments=FALSE, only non-ambiguous properly paired reads are counted. When fragments=TRUE, ambiguous reads are also counted (see "Pairing criteria" in readGAlignments()). fragments=TRUE is equivalent to the behavior of the TEtranscripts algorithm. For further details see summarizeOverlaps().

tolerance A positive numeric scalar storing the minimum tolerance above which the SQUAREM algorithm (Du and Varadhan, 2020) keeps iterating. Default is 1e-4 and this value is passed to the tol parameter of the squarem() function.

maxIter A positive integer scalar storing the maximum number of iterations of the SQUAREM algorithm (Du and Varadhan, 2020). Default is 100 and this value is passed to the maxiter parameter of the squarem() function.

References

Jin Y et al. TEtranscripts: a package for including transposable elements in differential expression analysis of RNA-seq datasets. Bioinformatics. 2015;31(22):3593-3599. DOI: https://doi.org/10.1093/bioinformatics/btv422

Jin Y et al. TEtranscripts: a package for including transposable elements in differential expression analysis of RNA-seq datasets. Bioinformatics. 2015;31(22):3593-3599. DOI: https://doi.org/10.1093/bioinformatics/btv422

Examples

```
bamfiles <- list.files(system.file("extdata", package="atena"),
                       pattern="*.bam", full.names=TRUE)
TE_annot <- readRDS(file = system.file("extdata", "Top28TEs.rds",
                    package="atena"))
ttpar <- TEtranscriptsParam(bamfiles, teFeatures = TE_annot,
                            singleEnd = TRUE, ignoreStrand=TRUE,
                            aggregateby = c("repName"))
ttpar
```

Index