Package ‘roar’

October 8, 2014

Type Package

Title Identify differential APA usage from RNA-seq alignments

Version 1.0.0

Date 2013-05-04

Author Elena Grassi

Maintainer Elena Grassi <grassi.e@gmail.com>

Description Identify preferential usage of APA sites, comparing two
biological conditions, starting from known alternative sites
and alignments obtained from standard RNA-seq experiments.

License GPL-3

Depends R (>= 3.0.1)

Imports GenomicRanges, GenomicAlignments(>= 0.99.4), methods,rtracklayer, IRanges

Suggests RUnit, BiocGenerics, RNAseqData.HNRNPC.bam.chr14

biocViews Sequencing, RNASeq, Transcription

R topics documented:

`roar-package` *Identify differential APA usage from RNA-seq alignments*

Description

Identify preferential usage of APA sites, comparing two biological conditions, starting from known alternative sites and alignments obtained from standard RNA-seq experiments.

Details

Package:	roar
Type:	Package
Version:	0.1.1
Date:	2013-05-04
License:	GPL-3

The codeRoarDataset class exposes methods to perform the whole analysis, in order to identify genes with preferential expression of long/short isoforms in a condition with respect to another one. The needed input data are alignments deriving from RNA-seq experiments of the two conditions and a set of coordinates of APA sites for genes with an alternative APA site proximal to the one used “normally”.

Author(s)

Elena Grassi <grassi.e@gmail.com>

`checkStep` *Private/inner/helper method to check the order of the invoked analysis methods*

Description

This method **should not** be used by package users. It gets an rds object and a required number of analysis step and, if possible, calls the requested method to reach that step. It returns the object and a logical value that tells if the analysis can go on.

Usage

```
checkStep(rds, neededStep)
```

Arguments

`rds`	A `RoarDataset` object.
`neededStep`	The analysis step where rds should be/arrive.

Value

A list containing a logical that shows if the needed step could be reached with rds and the object at the requested step. Check step won't repeat a step already done and the logical value will be FALSE in this case (and rds won't be returned modified).

`computePvals` *Computes pvalues (Fisher test) for this* `RoarDataset` *object*

Description

This is the third step in the Roar analyses: it applies a Fisher test comparing counts falling on the PRE and POST portion in the treatment and control conditions for every gene. If there are multiple samples for a condition every combinations of comparisons between the samples lists are considered.

Usage

```
computePvals(rds)
```

Arguments

`rds`	The `RoarDataset` which contains the counts over PRE-POST portions in the two conditions to be compared via pvalues.

Value

The `RoarDataset` object given as rds with the compute Roars phase of the analysis done. Pvalues will be held in the RoarDataset object itself in the case of single samples, while in a separate slot otherwise, but end user normally should not analyze those directly but use `totalResults` or `fpkmResults` at the end of the analysis.

Examples

```
library(GenomicAlignments)
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
rds <- countPrePost(rds, FALSE)
rds <- computeRoars(rds)
rds <- computePvals(rds)
```

`computeRoars` *Computes m/M and roar values for this* `RoarDataset` *object*

Description

This is the second step in the Roar analyses: it computes the ratio of prevalence of the short and long isoforms for every gene in the treatment and control condition (m/M) and their ratio, roar, that indicates if there is a relative shortening-lengthening in a condition over the other one. A roar > 1 for a given gene means that in the treatment condition that gene has an higher ratio of short vs long isoforms with respect to the control condition (and the opposite for roar < 1). Negative or NA m/M or roar occurs in not definite situations, such as counts equal to zero for PRE or POST portions. If for one of the conditions there are more than one samples then calculations are performed on average counts.

Usage

```
computeRoars(rds)
```

Arguments

`rds`	The `RoarDataset` which contains the counts over PRE-POST portions in the two conditions to be compared via roar.

Value

The `RoarDataset` object given as rds with the computeRoars phase of the analysis done. m/M and roars will be held in the RoarDataset object itself in the case of single samples, while in two slots otherwise, but end user normally should not analyze those directly but use `totalResults` or `fpkmResults` at the end of the analysis.

Examples

```
library(GenomicAlignments)
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
rds <- countPrePost(rds, FALSE)
rds <- computeRoars(rds)
```

---

cores — *Method to check how many cores are used by a roar analysis - right now not useful*

Description

Right now always returns 1 as long as multi-core support has to be implemented.

Usage

```
cores(rds)
```

Arguments

rds	A RoarDataset object.

Value

The number of cores used by this roar analisys.

countPrePost *Counts reads falling over PRE/POST portions of the given transcripts*

Description

This is the first step in the Roar analyses: it counts reads overlapping with the PRE/POST portions defined in the given gtf/GRanges annotation. See RoarDataset for details on how to define these portions. Reads of the given bam annotation files that falls over this portion are accounted for with the following rules:

1- reads that align on only one of the given features are assigned to that feature, even if the overlap is not complete 2- reads that align on both a PRE and a POST feature of the same gene (spanning reads) are assigned to the POST one, considering that they have clearly been obtained from the longer isoform

If the stranded argument is set to TRUE then strandness is considered when counting reads.

Usage

```
countPrePost(rds, stranded=FALSE)
```

Arguments

rds	The RoarDataset which contains the alignments and annotation informations over which counts will be performed.
stranded	A logical indicating if strandness should be considered when counting reads or not. Default=FALSE.

Value

The RoarDataset object given as rds with the counting reads phase of the analysis done. Counts will be held in the RoarDataset object itself in the case of single samples, while in two slots otherwise, but end user normally should not analyze those directly.

Examples

```
 library(GenomicAlignments)
 gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
 features <- GRanges(
    seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
    strand = strand(rep("+", length(gene_id))),
    ranges = IRanges(
       start=c(1000, 2000, 3000, 3600),
       width=c(1000, 900, 600, 300)),
    DataFrame(gene_id)
 )
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
```

```
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
 rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
 rds <- countPrePost(rds)
```

`countResults` *Returns a dataframe with results of the analysis for a* `RoarDataset` *object*

Description

The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (number of reads falling over the PRE portions).

Usage

```
countResults(rds)
```

Arguments

`rds`	The `RoarDataset` with all the analysis steps (`countPrePost`, `computeRoars`, `computePvals`) performed. If one or more steps hadn't been performed they will be called automatically.

Value

The resulting dataframe will be identical to that returned by `link{totalResults}` but with two columns added: "treatmentValue" and "controlValue". These columns will contain a number that indicates the level of expression of the relative gene in the treatment (or control) condition. This number represents the counts (averages across samples when applicable) obtained for the PRE portion of the gene and is similar. See the vignette for more details.

Examples

```
 library(GenomicAlignments)
 gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
 features <- GRanges(
    seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
    strand = strand(rep("+", length(gene_id))),
    ranges = IRanges(
       start=c(1000, 2000, 3000, 3600),
       width=c(1000, 900, 600, 300)),
    DataFrame(gene_id)
 )
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
 rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
 rds <- countPrePost(rds, FALSE)
 rds <- computeRoars(rds)
 rds <- computePvals(rds)
 dat <- countResults(rds)
```

`fpkmResults` *Returns a dataframe with results of the analysis for a* `RoarDataset` *object*

Description

The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (a measure recalling FPKM).

Usage

```
fpkmResults(rds)
```

Arguments

`rds`	The `RoarDataset` with all the analysis steps (`countPrePost`, `computeRoars`, `computePvals`) performed. If one or more steps hadn't been performed they will be called automatically.

Value

The resulting dataframe will be identical to that returned by `link{totalResults}` but with two columns added: "treatmentValue" and "controlValue". These columns will contain a number that indicates the level of expression of the relative gene in the treatment (or control) condition. This number derives from the counts (averages across samples when applicable) obtained for the PRE portion of the gene and is similar to the FPKM standard measure of expression deriving from RNAseq experiment. See the vignette for more details.

Examples

```
library(GenomicAlignments)
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
```

```
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
 rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
 rds <- countPrePost(rds, FALSE)
 rds <- computeRoars(rds)
 rds <- computePvals(rds)
 dat <- fpkmResults(rds)
```

---

getFisher *Private/inner/helper method to perform Fisher test*

Description

This method **should not** be used by package users. Given a numerical vector of length 4 it will perform a Fisher test and return the p-value for the two.sided test. Non-integer values will be rounded.

Usage

```
getFisher(counts)
```

Arguments

counts	A numerical vector of length 4.

Value

The pvalue for the two.sided Fisher test.

---

meanAcrossAssays *Private/inner/helper method to get average counts across samples*

Description

This method **should not** be used by package users. It gets average counts for "pre" or "post" portions (depending on the wantedColumns argument) given the list of assays for one of the two conditions.

Usage

```
meanAcrossAssays(assays, wantedColumns)
```

Arguments

`assays`	A list of matrixes/dataframes.
`wantedColumns`	The name of the columns ("pre" or "post") whose means should be computed. Average will be calculated on the corresponding rows of the list of matrixes/dataframe, working on the given column.

Value

The pvalue for the two.sided Fisher test.

`pvalueFilter` *Returns a dataframe with results of the analysis for a* `RoarDataset` *object*

Description

The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (a measure recalling FPKM). Only the genes with an expression estimate bigger than a given cutoff will be considered. Also pvalues will be considered for filtering.

Usage

```
pvalueFilter(rds, fpkmCutoff, pvalCutoff)
```

Arguments

`rds`	The `RoarDataset` with all the analysis steps (`countPrePost`, `computeRoars`, `computePvals`) performed. If one or more steps hadn't been performed they will be called automatically.
`fpkmCutoff`	The cutoff that will be used to determine if a gene is expressed or not.
`pvalCutoff`	The cutoff that will be used to determine if a pvalue is significative or not.

Value

The resulting dataframe will be identical to that returned by `link{standardFilter}` but after gene expression filtering and pvalue correction another step will be performed: for single samples comparisons only genes with a nominal pvalue smaller than the given cutoff will be considered, while for multiple samples a column (nUnderCutoff) will be added to the dataframe. This column will contain an integer number representing the number of comparisons between the samples of the two conditions that results in a nominal pvalue lower than the given cutoff (pvalCutoff).

Examples

```
library("GenomicAlignments")
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
rds <- countPrePost(rds, FALSE)
rds <- computeRoars(rds)
rds <- computePvals(rds)
dat <- pvalueFilter(rds, 1, 0.05)
```

`RoarDataset` *Creates a* `RoarDataset` *object*

Description

This function creates an `RoarDataset` object from two lists of of `GAlignments` and a `GRanges` containing a suitable annotation of alternative APA sites.

Usage

```
RoarDataset(treatmentGappedAlign, controlGappedAlign, gtfGRanges)
```

Arguments

`treatmentGappedAlign`
: A list of `GAlignments` representing alignment of samples for the treatment condition (by convention it is considered the “treated” condition: this simply means that the package will compute roar values (ratios of the m/M) using this condition as the numerator) to be considered.

`controlGappedAlign`
: A list of `GAlignments` representing alignment of samples for the control condition to be considered.

`gtfGRanges`
: A `GRanges` object with coordinates for the portions of transcripts that has to be considered pertaining to the short (or long) isoform. This `GRanges` object must have a character elementMetadata called "gene_id" that ends with "_PRE" or "_POST" to address respectively the short and the long isoform. An element in

the annotation is considered "PRE" (i.e. common to the short and long isoform of the transcript) if its gene_id ends with "_PRE". If it ends with "_POST" it is considered the portion present only in the long isoform. The prefix of gene_id should be a unique identifier for the gene and each identifier has to be associated with only one "_PRE" and one "_POST", leading to two genomic region associated to each gene_id.

Value

A `RoarDataset` object ready to be analyzed via the other methods.

See Also

`RoarDatasetFromFiles`

Examples

```
 library(GenomicAlignments)
 gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
 features <- GRanges(
    seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
    strand = strand(rep("+", length(gene_id))),
    ranges = IRanges(
       start=c(1000, 2000, 3000, 3600),
       width=c(1000, 900, 600, 300)),
    DataFrame(gene_id)
 )
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
 rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
```

RoarDataset-class *Class* "RoarDataset"

Description

RoarDataset - a class to perform 3'UTR shortening analyses

Objects from the Class

Objects of thiss class should be created using the functions `RoarDataset` or `RoarDatasetFromFiles`, ideally the raw `new` method should never be invoked by end users. Then to perform the analysis the user should call, in order: countPrePost, computeRoars, computePvals and one of the methods to format results.

Slots

`treatmentBams`: Object of class `"list"` - a list of GappedAlignment objects for the first condition (by convention it is considered the “treated” condition) in analysis.

`controlBams`: Object of class `"list"` - a list of GappedAlignment objects for the second condition (by convention it is considered the “control” condition) in analysis.

`prePostCoords`: Object of class `"GRanges"` - represents the APA sites coords, defining "PRE" (last exon coords up until the alternative APA, defining the shorter isoform) and "POST" (from the alternative APA to the “standard” one) regions of the genes.

`postCoords`: Object of class `"GRanges"` - private object.

`countsTreatment`: Object of class `"SummarizedExperiment"` - private object.

`countsControl`: Object of class `"SummarizedExperiment"` - private object.

`pVals`: Object of class `"SummarizedExperiment"` - private object.

`step`: Object of class `"numeric"` - private object.

`cores`: Object of class `"numeric"` - private object.

`exptData`: Object of class `"SimpleList"` - private object.

`rowData`: Object of class `"GenomicRangesORGRangesList"` - private object.

`colData`: Object of class `"DataFrame"` - private object.

`assays`: Object of class `"Assays"` - private object.

Extends

Class `"SummarizedExperiment"`, directly.

Methods

`countPrePost` `signature(rds = "RoarDataset", stranded = "logical")`: Counts reads falling over PRE/POST portions of the given transcripts.

`computeRoars` `signature(rds = "RoarDataset")`: Computes m/M and roar values for this `RoarDataset` object.

`computePvals` `signature(rds = "RoarDataset")`: Computes pvalues (Fisher test) for this `RoarDataset` object.

totalResults `signature(rds = "RoarDataset")`: Returns a dataframe with results of the analysis for a `RoarDataset` object.

`fpkmResults` `signature(rds = "RoarDataset")`: The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (a measure recalling FPKM).

`countResults` `signature(rds = "RoarDataset")`: The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (counts over PRE portions).

`standardFilter` `signature(rds = "RoarDataset", fpkmCutoff = "double")`: Returns a dataframe with results of the analysis for a `RoarDataset` object.

`pvalueFilter` `signature(rds = "RoarDataset", fpkmCutoff = "double", pvalCutoff = "double")`: ...

`cores` `signature(rds = "RoarDataset")`: returns the number of cores used for computation, right now always 1.

Author(s)

Elena Grassi, PhD student in Biomedical Sciences and Oncology - Dept. of Molecular Biotechnologies and Health Sciences, Molecular Biotechnology Center, Torino

Examples

```
showClass("RoarDataset")
```

RoarDatasetFromFiles *Creates a RoarDataset object*

Description

This function creates an RoarDataset object from two list and a gtf with a suitable annotation of alternative APA sites.

Usage

```
        RoarDatasetFromFiles(treatmentBams, controlBams, gtf)
```

Arguments

treatmentBams	A list of filenames of bam alignments with data for the treatment condition (by convention it is considered the “treated” condition: this simply means that the package will compute roar values (ratios of the m/M) using this condition as the numerator) to be considered.
controlBams	A list of filenames of bam alignments with data for the control condition to be considered.
gtf	A filename of a gtf with coordinates for the portions of transcripts that has to be considered pertaining to the short (or long) isoform. This gtf must have an attribute called "gene_id" that ends with "_PRE" or "_POST" to address respectively the short and the long isoform. A ready-to-go gtf, with coordinates derived from the PolyADB on the human genome (version hg19), is available in the "examples" package directory. An element in the annotation is considered "PRE" (i.e. common to the short and long isoform of the transcript) if its gene_id feature in the gtf ends with "_PRE". If it ends with "_POST" it is considered the portion present only in the long isoform. The prefix of gene_id should be an identifier for the gene and each identifier has to be associated with only one "_PRE" and one "_POST", leading to two genomic region associated to each gene_id.

Value

A RoarDataset object ready to be analyzed via the other methods.

See Also

RoarDataset

Examples

```
#rds <- RoarDatasetFromFiles(treatmentBams, controlBams, gtf)
```

---

standardFilter *Returns a dataframe with results of the analysis for a* RoarDataset *object*

Description

The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (a measure recalling FPKM). Only the genes with an expression estimate bigger than a given cutoff will be considered.

Usage

```
standardFilter(rds, fpkmCutoff)
```

Arguments

rds	The RoarDataset with all the analysis steps (countPrePost, computeRoars, computePvals) performed. If one or more steps hadn't been performed they will be called automatically.
fpkmCutoff	The cutoff that will be used to determine if a gene is expressed or not.

Value

The resulting dataframe will be identical to that returned by `link{fpkmResults}` but it will contains rows relative only with genes with an expression estimate (treatment or controlValue) bigger than the given fpkmCutoff in both the conditions.

Examples

```
library(GenomicAlignments)
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
```

```
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
 rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
 rds <- countPrePost(rds, FALSE)
 rds <- computeRoars(rds)
 rds <- computePvals(rds)
 dat <- standardFilter(rds, 1)
```

totalResults *Returns a dataframe with results of the analysis for a* RoarDataset *object*

Description

The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values and pvalues.

Usage

```
totalResults(rds)
```

Arguments

rds	The RoarDataset with all the analysis steps (countPrePost, computeRoars, computePvals) performed.

Value

The RoarDataset object given as rds with all the analysis steps performed. If one or more steps hadn't been performed they will be called automatically. The resulting dataframe will have the "gene_id" of the initial annotation as row names (without the trailing "_PRE"/"_POST") and as columns the m/M ratio for the treatment and control conditions, the roar value and the Fisher test pvalue (respectively: mM_treatment, mM_control, roar, pval). If more than one sample has been given for a condition the "pval" column will contain the multiplication of all the comparisons pvalue and there will be other columns containing the pvalues resulting from all the pairwise treatment vs control contrasts, with names "pvalue_X_Y" where X represent the position of the sample in the treatment list of bam files (or GappedAlignment) and Y the position for the control list. **WARNING**: this method does not filter in any way the results, therefore there will be negative m/M values/ROAR and also NA - in these cases there aren't enough information to draw a conclusion about the shortening/lengthening of the gene in the given samples and thus the pvalues should not be kept in consideration. Furthermore there isn't any filter on the expression level of the genes. See fpkmResults, standardFilter and pvalueFilter about results filtering possibilities.

Examples

```
library(GenomicAlignments)
gene_id <- c("A_PRE", "A_POST", "B_PRE", "B_POST")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr1", "chr2", "chr2")),
   strand = strand(rep("+", length(gene_id))),
   ranges = IRanges(
      start=c(1000, 2000, 3000, 3600),
      width=c(1000, 900, 600, 300)),
   DataFrame(gene_id)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rd2 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(2000), cigar = "300M", strand = strand("+"))
rd3 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3000), cigar = "300M", strand = strand("+"))
rds <- RoarDataset(list(c(rd1,rd2)), list(rd3), features)
rds <- countPrePost(rds, FALSE)
rds <- computeRoars(rds)
rds <- computePvals(rds)
dat <- totalResults(rds)
```

Index