Package 'roar'

November 12, 2026

Type Package

Title Identify differential APA usage from RNA-seq alignments

Version 1.48.0

Date 2016-03-21

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.

biocViews Sequencing, HighThroughputSequencing, RNAseq, Transcription

License GPL-3

Depends R (>= 3.0.1)

Imports methods, BiocGenerics, S4Vectors, IRanges, GenomicRanges, SummarizedExperiment, GenomicAlignments (>= 0.99.4), rtracklayer, GenomeInfoDb

Suggests RNAseqData.HNRNPC.bam.chr14, testthat

URL https://github.com/vodkatad/roar/

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

git_branch RELEASE_3_23

git_last_commit fb71970

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

roar-package 2
checkStep 3
combineFisherMethod 3
computePairedPvals 4

computePvals 5
computeRoars 6
cores 7
countPrePost 8
countResults 9
fpkmResults 10
getFisher 11
meanAcrossAssays 12
pvalueCorrectFilter 12
pvalueFilter 14
RoarDataset 15
RoarDataset-class 16
RoarDatasetFromFiles 18
RoarDatasetMultipleAPA 19
RoarDatasetMultipleAPA-class 20
RoarDatasetMultipleAPAFromFiles 22
standardFilter 23
totalResults 24

Index **26**

`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

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).

combineFisherMethod — *Private/inner/helper method to combine pvalues of independent test with the Fisher method*

Description

This method **should not** be used by package users. Given a numerical vector of pvalues, which should be obtained from independent tests on the same null hypothesis, this will give the combined pvalue following the Fisher method.

Usage

```
combineFisherMethod(pvals)
```

Arguments

pvals	A numerical vector with pvalues of independent tests on the same H0.

Value

The combined pvalue given by the Fisher method.

computePairedPvals *Computes pvalues (Fisher test) on the read counts in this roar analysis*

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. The paired method should be used when the experimental setup offers multiple paired samples for the two conditions: that is foreach sample of the control condition there is a naturally paired one for the treatment (i.e. cells derived from the same plate divided in two groups and treated or not). For example in the below code sample treatment sample n.1 (rd1) is paired with control n.2 (rd4) and rd2 with rd3. The pvalue resulting from Fisher test applied on the different samples pairings will be combined with the Fisher method, therefore the pairs of samples should be independent between each other.

Usage

```
computePairedPvals(rds, treatmentSamples, controlSamples)
```

Arguments

rds	The RoarDataset or the RoarDatasetMultipleAPA which contains the counts over PRE-POST portions in the two conditions to be compared via pvalues.
treatmentSamples	Numbers that represent the indexes of the treatmentBams/GappedAlign parameter given to the RoarDataset costructor and the order in which they are paired with control samples.
controlSamples	Numbers that represent the indexes of the controlBams/GappedAlign parameter given to the RoarDataset costructor and the order in which they are paired with treatment samples.

Value

The RoarDataset or the RoarDatasetMultipleAPA object given as rds with the compute pvalues 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("+"))
rd4 <- GAlignments("a", seqnames = Rle("chr2"), pos = as.integer(3400), cigar = "300M", strand = strand("+"))
rds <- RoarDataset(list(rd1,rd2), list(rd3, rd4), features)
rds <- countPrePost(rds, FALSE)
rds <- computeRoars(rds)
rds <- computePairedPvals(rds, c(1,2), c(2,1))
```

`computePvals` *Computes pvalues (Fisher test) on the read counts in this roar analysis*

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` or the `RoarDatasetMultipleAPA` which contains the counts over PRE-POST portions in the two conditions to be compared via pvalues.

Value

The `RoarDataset` or the `RoarDatasetMultipleAPA` object given as rds with the compute pvalue 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*

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, qwidthTreatment=NA, qwidthControl=NA)
computeRoars(rds, qwidthTreatment, qwidthControl)
```

Arguments

`rds`	The `RoarDataset` or the `RoarDatasetMultipleAPA` which contains the counts over PRE-POST portions in the two conditions to be compared via roar.
`qwidthTreatment`	The mean length of the reads in the treatment bam files - used internally for the interaction between `RoarDataset` and `RoarDatasetMultipleAPA` objects. The default (NA) calculates this value from the bam and should not be changed.
`qwidthControl`	The mean length of the reads in the control bam files - used internally for the interaction between `RoarDataset` and `RoarDatasetMultipleAPA` objects. The default (NA) calculates this value from the bam and should not be changed.

Value

The RoarDataset or the RoarDatasetMultipleAPA 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 longest isoform

If the stranded argument is set to TRUE then strandness is considered when counting reads. When rds is a RoarDatasetMultipleAPA counts are obtained on more than two portions for each transcript in order to be able to efficiently evaluate multiple APA sites. The option stranded=TRUE is still not implemented for RoarDatasetMultipleAPA.

Usage

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

Arguments

rds	The RoarDataset or the RoarDatasetMultipleAPA 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. **WARNING**: not implemented (ignored) when using RoarDatasetMultipleAPA.

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

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 or a* RoarDatasetMultipleAPA *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 or the RoarDatasetMultipleAPA 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. For RoarDataset this number represents the counts (averaged across samples when applicable) obtained for the PRE portion of the gene. For RoarDatasetMultipleAPA every possible PRE choice will have its corresponding reads counts assigned and also the length of the PRE portion (counting only exonic bases). 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 or a* `RoarDatasetMultipleAPA` *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` or the `RoarDatasetMultipleAPA` 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 `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. For `RoarDataset` this number derives from the counts (averages across samples when applicable) obtained for the PRE portion of the gene and is similar to the RPKM standard measure of expression used in RNAseq experiment. Specifically we correct the counts on the PRE portions dividing them by portion length and total numer of reads aligned on all PRE portions and the multiply the results for 1000000000. See the vignette for more details.

For `RoarDatasetMultipleAPA` the same procedure is applied to all the possible PRE choices for genes. Note that summing all the counts for every PRE portion assigned to a gene could lead to count some reads multiple times when summing all the PRE portions counts therefore this measure is not completely comparable with the one obtained with the single APA analysis. The length column added in this case contains the length of the PRE portions (counting only exonic bases).

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 matrices/dataframe, working on the given column.

Value

The pvalue for the two.sided Fisher test.

`pvalueCorrectFilter` *Returns a dataframe with results of the analysis for a* `RoarDataset` *object or a* `RoarDatasetMultipleAPA` *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, corrected considering multiple testing, will be considered for filtering.

Usage

```
pvalueCorrectFilter(rds, fpkmCutoff, pvalCutoff, method)
```

Arguments

rds	The RoarDataset or the RoarDatasetMultipleAPA 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.
method	The multiple test correction method that has to be used (used only for multiple paired samples or single samples, not used for multiple unpaired samples.)

Value

For RoarDataset:

The resulting dataframe will be identical to that returned by standardFilter but after gene expression filtering another step will be performed: for single samples comparisons or multiple paired samples comparisons only genes with a corrected (with the given method) pvalue (for paired datasets this is the combined pvalue obtained with the Fisher method) smaller than the given cutoff will be returned, 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).

For RoarDatasetMultipleAPA: for each gene we select the APA choice that is associated with the smallest p-value then proceed exactly as above.

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)
```

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

For RoarDataset:

The resulting dataframe will be identical to that returned by standardFilter but after gene expression and m/M values filtering 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). For multiple samples with a paired design (i.e. if computePairedPvals was used) the pvalues of the requested pairings will be listed together with the combined pvalued obtained with the Fisher method and the filtering will be done on this pvalue.

For RoarDatasetMultipleAPA: for each gene we select the APA choice that is associated with the smallest p-value then proceed exactly as above.

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 metadata column 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. The GRanges object can also contain a numeric metadata column that represents the lengths of PRE and POST portions on the transcriptome. If this is omitted the lengths on the genome are used instead. Note that right now every gtf entry (or none of them) should have it.

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 `"RangedSummarizedExperiment"` - private object.

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

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

`paired`: `"logical"` slot - private.

`step`: `"numeric"` slot - private.

`cores`: `"numeric"` slot - private.

`metadata`: `"list"` slot - private.

`rowRanges`: Object of class `"GRangesORGRangesList"` - private object.

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

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

Extends

Class `"RangedSummarizedExperiment"`, 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 lists 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. The gtf can also contain an attribute that represents the lengths of PRE and POST portions on the transcriptome. If this is omitted the lengths on the genome are used instead. Note that right now every gtf entry (or none of them) should have it.

Value

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

See Also

`RoarDataset`

Examples

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

RoarDatasetMultipleAPA

Creates a RoarDatasetMultipleAPA *object*

Description

This function creates an RoarDatasetMultipleAPA object from two lists of of GAlignments and a GRanges containing a suitable annotation of alternative APA sites and gene exon structure. A MultipleAPA analysis computes several roar values and p-values for each gene: one for every possible combination of APA-canonical end of a gene (i.e. the end of its last exon). This is more efficient than performing several different "standard" roar analyses choosing the PRE and POST portions corresponding to different APAs because reads overlaps are computed only once.

Usage

```
RoarDatasetMultipleAPA(treatmentBamsGenomicAlignments, controlBamsGenomicAlignments, gtfGRanges)
```

Arguments

treatmentBamsGenomicAlignments	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.
controlBamsGenomicAlignments	A list of GAlignments representing alignment of samples for the control condition to be considered.
gtfGRanges	A GRanges containing a suitable annotation of alternative APA sites and gene exonic structure. Minimal requirements are: metadata columns called "gene", "apa" and "type." APA should be single bases falling over one of the given genes and need to have the metadata column "type" equal to "apa" and the "apa" column composed of unambiguous id and the corresponding gene id pasted together with an underscore. The "gene" metadata columns for these entries should not be initialized. All the studied gene exons need to be reported, in this case the metadata column "gene" should contain the gene id (the same one reported for each gene APAs) while "type" should be set to "gene" and "apa" to NA. All apa entries assigned to a gene should have coordinates that falls inside it and every gene that appears should contain at least one APA.

Value

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

See Also

RoarDatasetMultipleAPAFromFiles

Examples

```
library(GenomicAlignments)
gene <- c("A", "B", NA, NA)
type <- c("gene","gene","apa", "apa")
apa <- c(NA, NA, "apa1_A", "apa2_B")
features <- GRanges(
   seqnames = Rle(c("chr1", "chr2", "chr1", "chr2")),
   strand = strand(rep("+", length(gene))),
   ranges = IRanges(
      start=c(1000, 2000, 1300, 2050),
      width=c(500, 900, 1, 1)),
   DataFrame(gene, apa, type)
)
rd1 <- GAlignments("a", seqnames = Rle("chr1"), pos = as.integer(1000), cigar = "300M", strand = strand("+"))
rds <- RoarDatasetMultipleAPA(list(c(rd1,rd1)), list(c(rd1,rd1)), features)
```

RoarDatasetMultipleAPA-class

Class "RoarDatasetMultipleAPA"

Description

RoarDataset - a class to perform 3'UTR shortening analyses

Objects from the Class

Objects of thiss class should be created using the functions RoarDatasetMultipleAPA or RoarDatasetMultipleAPAFromFi 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. This class is used to allow efficient analyses that allow to study more than one APA site for each gene: internally it uses a RoarDataset object that stores PRE/POST counts for all possible alternative APA choices for each gene.

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.

`geneCoords`: Object of class "GRangesList" - private object that represents the exon structures of genes in study.

`apaCoords`: Object of class `"GRangesList"` - private object that represents the APA fallin on genes in study.

`fragments`: Object of class `"GRangesList"` - private object used to efficiently count reads falling on short and long isoforms.

`prePostDef`: Object of class `"list"` - private object representing all possible short and long isoforms.

`roars`: Object of class `"list"` - private object with a list of `RoarDataset` objects, each one representing all possible PRE/POST choices for a single gene.

`corrTreatment`: `"numeric"` slot - private, integer representing the mean length of reads for the treatment samples.

`corrControl`: `"numeric"` slot - private, integer representing the mean length of reads for the control samples.

`paired`: `"logical"` slot - private.

`step`: `"numeric"` slot - private.

`cores`: `"numeric"` slot - private.

Methods

`countPrePost` `signature(rds = "RoarDatasetMultipleAPA", stranded = "logical")`: Counts reads falling over all the possible PRE/POST portions of the given transcripts. WARNING: stranded = TRUE is still unsupported and could give unpredictable results.

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

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

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

`fpkmResults` `signature(rds = "RoarDatasetMultipleAPA")`: 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 = "RoarDatasetMultipleAPA")`: The last step of a classical Roar analyses: it returns a dataframe containing m/M values, roar values, pvalues and estimates of expression (counts of reads falling over a gene).

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

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

`cores` `signature(rds = "RoarDatasetMultipleAPA")`: 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("RoarDatasetMultipleAPA")
```

RoarDatasetMultipleAPAFromFiles

Creates a RoarDatasetMultipleAPA *object*

Description

This function creates an RoarDatasetMultipleAPA object from two lists and a gtf with a suitable annotation of alternative APA sites and exonic structures of genes. A MultipleAPA analysis computes several roar values and p-values for each gene: one for every possible combination of APA-canonical end of a gene (i.e. the end of its last exon). This is more efficient than performing several different "standard" roar analyses choosing the PRE and POST portions corresponding to different APAsbecause reads overlaps are computed only once.

Usage

```
RoarDatasetMultipleAPAFromFiles(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 alternative APA sites and gene exonic structure. This gtf must have three attributes called "gene", "apa" and "type" to distinguish different features. APA should be single bases falling over one of the given genes and need to have the attribute "type" equal to "apa" and the "apa" attribute composed of unambiguous id and the corresponding gene id pasted together with an underscore. The "gene" attributes for these entries should not be initialized. All the studied gene exons need to be reported, in this case the attribute "gene" should contain the gene id (the same one reported for each gene APAs) while "type" should be set to "gene" and "apa" to NA. All apa entries assigned to a gene should have coordinates that falls inside it and every gene that appears should contain at least one APA. A ready-to-go gtf, with coordinates derived from the PolyADB on the human genome (version hg19), is available in the "examples" package directory.

Value

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

See Also

RoarDatasetMultipleAPA

Examples

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

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

For RoarDataset and RoarDatasetMultipleAPA:

The resulting dataframe will be identical to that returned by 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 and with sensitive m/M and roar values (it removes negative or NA m/M values/roar - these values arise when there aren't enough information to draw a conclusion about the shortening/lengthening of the gene).

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` *or a* `RoarDatasetMultipleAPA` *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` or `RoarDatasetMultipleAPA` with all the analysis steps (`countPrePost`, `computeRoars`, `computePvals`) performed.

Value

The `RoarDataset` or the `RoarDatasetMultipleAPA` 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 product 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. When using `RoarDatasetMultipleAPA` this dataframe will report multiple results for each gene that corresponds to the pairings between every APA associated with that gene in the gtf and the gene's end - rownames in this case will be in the form geneid_apaid. **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

∗ **RoarDatasetFromFiles**
RoarDatasetFromFiles, 18
RoarDatasetMultipleAPAFromFiles, 22
∗ **RoarDataset**
RoarDataset, 15
RoarDatasetMultipleAPA, 19
∗ **checkStep**
checkStep, 3
cores, 7
∗ **classes**
RoarDataset-class, 16
RoarDatasetMultipleAPA-class, 20
∗ **combineFisherMethod**
combineFisherMethod, 3
∗ **computePairedPvals**
computePairedPvals, 4
∗ **computePvals**
computePvals, 5
∗ **computeRoars**
computeRoars, 6
∗ **countPrePost**
countPrePost, 8
∗ **countResults**
countResults, 9
∗ **fpkmResults**
fpkmResults, 10
∗ **getFisher**
getFisher, 11
∗ **meanAcrossAssays**
meanAcrossAssays, 12
∗ **package**
roar-package, 2
∗ **pvalueCorrectFilter**
pvalueCorrectFilter, 12
∗ **pvalueFilter**
pvalueFilter, 14
∗ **standardFilter**
standardFilter, 23
∗ **totalResults**
totalResults, 24

checkStep, 3
combineFisherMethod, 3
computePairedPvals, 4
computePairedPvals, RoarDataset *(computePairedPvals)*, 4
computePairedPvals, RoarDatasetMultipleAPA *(computePairedPvals)*, 4
computePairedPvals,RoarDataset,numeric,numeric-method *(RoarDataset-class)*, 16
computePairedPvals,RoarDatasetMultipleAPA,numeric,numeric-*(RoarDatasetMultipleAPA-class)*, 20
computePvals, 5, *9, 10*, *13, 14*, *17*, *21*, *23, 24*
computePvals, RoarDataset *(computePvals)*, 5
computePvals, RoarDatasetMultipleAPA *(computePvals)*, 5
computePvals,RoarDataset-method *(RoarDataset-class)*, 16
computePvals,RoarDatasetMultipleAPA-method *(RoarDatasetMultipleAPA-class)*, 20
computeRoars, 6, *9, 10*, *13, 14*, *17*, *21*, *23, 24*
computeRoars, RoarDataset, numeric, numeric *(computeRoars)*, 6
computeRoars, RoarDatasetMultipleAPA, numeric, numeric *(computeRoars)*, 6
computeRoars,RoarDataset-method *(RoarDataset-class)*, 16
computeRoars,RoarDatasetMultipleAPA-method *(RoarDatasetMultipleAPA-class)*, 20
cores, 7, *17*, *21*
cores,RoarDataset-method *(RoarDataset-class)*, 16

cores,RoarDatasetMultipleAPA-method (RoarDatasetMultipleAPA-class), 20
countPrePost, 8, *9, 10*, *13, 14*, *17*, *21*, *23, 24*
countPrePost,RoarDataset,logical (countPrePost), 8
countPrePost,RoarDataset,logical-method (RoarDataset-class), 16
countPrePost,RoarDataset-method (RoarDataset-class), 16
countPrePost,RoarDatasetMultipleAPA (countPrePost), 8
countPrePost,RoarDatasetMultipleAPA,logical-method (RoarDatasetMultipleAPA-class), 20
countPrePost,RoarDatasetMultipleAPA-method (RoarDatasetMultipleAPA-class), 20
countResults, 9, *17*, *21*
countResults, RoarDataset (countResults), 9
countResults, RoarDatasetMultipleAPA (countResults), 9
countResults,RoarDataset-method (RoarDataset-class), 16
countResults,RoarDatasetMultipleAPA-method (RoarDatasetMultipleAPA-class), 20

fpkmResults, *4, 5*, *7*, 10, *17*, *21*, *23*, *25*
fpkmResults, RoarDataset (fpkmResults), 10
fpkmResults, RoarDatasetMultipleAPA (fpkmResults), 10
fpkmResults,RoarDataset-method (RoarDataset-class), 16
fpkmResults,RoarDatasetMultipleAPA-method (RoarDatasetMultipleAPA-class), 20

GAlignments, *15*, *19*
getFisher, 11
GRanges, *15*, *19*

meanAcrossAssays, 12

new, *16*, *20*

pvalueCorrectFilter, 12
pvalueCorrectFilter, RoarDataset (pvalueCorrectFilter), 12
pvalueCorrectFilter, RoarDatasetMultipleAPA (pvalueCorrectFilter), 12
pvalueCorrectFilter,RoarDataset,numeric,numeric,character-(RoarDataset-class), 16
pvalueCorrectFilter,RoarDatasetMultipleAPA,numeric,numeric (RoarDatasetMultipleAPA-class), 20
pvalueFilter, 14, *17*, *21*, *25*
pvalueFilter, RoarDataset (pvalueFilter), 14
pvalueFilter, RoarDatasetMultipleAPA (pvalueFilter), 14
pvalueFilter,RoarDataset,numeric,numeric-method (RoarDataset-class), 16
pvalueFilter,RoarDatasetMultipleAPA,numeric,numeric-method (RoarDatasetMultipleAPA-class), 20

RangedSummarizedExperiment, *17*
roar (roar-package), 2
roar-package, 2
RoarDataset, *2–10*, *12–15*, 15, *16–18*, *20, 21*, *23, 24*
RoarDataset-class, 16
RoarDatasetFromFiles, *16*, 18
RoarDatasetMultipleAPA, *4–10*, *12–14*, *19*, 19, *20–24*
RoarDatasetMultipleAPA-class, 20
RoarDatasetMultipleAPAFromFiles, *20*, 22

standardFilter, *13, 14*, *17*, *21*, 23, *25*
standardFilter, RoarDataset (standardFilter), 23
standardFilter, RoarDatasetMultipleAPA (standardFilter), 23
standardFilter,RoarDataset,numeric-method (RoarDataset-class), 16
standardFilter,RoarDatasetMultipleAPA,numeric-method (RoarDatasetMultipleAPA-class), 20

totalResults, *4, 5*, *7*, *10*, 24
totalResults, RoarDataset (totalResults), 24

totalResults, RoarDatasetMultipleAPA
(totalResults), 24
totalResults,RoarDataset-method
(RoarDataset-class), 16
totalResults,RoarDatasetMultipleAPA-method
(RoarDatasetMultipleAPA-class),
20