Package 'easyRNASeq'

April 5, 2014

Version 1.8.8

Date 2014-04-03

Type Package

Title Count summarization and normalization for RNA-Seq data.

Author Nicolas Delhomme, Ismael Padioleau, Bastian Schiffthaler

Maintainer Nicolas Delhomme <delhomme@embl.de>

Description Calculates the coverage of high-throughput short-reads against
a genome of reference and summarizes it per feature of interest (e.g. exon,gene, transcript). The data can be normalized as 'RPKM' or by the 'DESeq'
or 'edgeR' package.

Depends genomeIntervals (>= 1.18.0), Biobase (>= 2.22.0), biomaRt (>=
2.18.0), edgeR (>= 3.4.0), Biostrings (>= 2.30.0), DESeq (>= 1.14.0), GenomicRanges (>= 1.14.3), IRanges (>= 1.20.5),Rsamtools (>= 1.14.1), ShortRead (>= 1.20.0)

Imports graphics, methods, parallel, utils, BiocGenerics (>= 0.8.0),LSD (>= 2.5)

Suggests BSgenome (>= 1.30.0), BSgenome.Dmelanogaster.UCSC.dm3 (>=
1.3.19), GenomicFeatures (>= 1.14.0), RnaSeqTutorial (>= 0.0.13), BiocStyle (>= 1.0.0)

License Artistic-2.0

LazyLoad yes

biocViews GeneExpression, RNAseq, Genetics, Preprocessing

R topics documented:

`count,character-method` *count method*

Description

This function is to supersed the easyRNASeq function in order to consolidate the option parameters as well as the option output. Ideally, the only output would be a SummarizedExperiment.

Usage

```
## S4 method for signature character
count(filesDirectory = getwd(),
  outputFormat = "SummarizedExperiment", ...)
```

Arguments

`filesDirectory`	The directory where the files to be used are located.
`outputFormat`	By default, easyRNASeq returns a `SummarizedExperiment`. If one of `DESeq`,`edgeR`,`RNAseq`, `matrix` is provided then the respective object is returned. Ideally, this option should get deprecated and only a SummarizedExperiment returned.
`...`	currently additional arguments to the easyRNASeq function.

Value

Returns a `SummarizedExperiment`. If the `outputFormat` option has been set, a corresponding object is returned: a count table (a matrix of m features x n samples), a `DESeq:newCountDataset`, a `edgeR:DGEList` or `RNAseq`.

Author(s)

Nicolas Delhomme

See Also

RNAseq SummarizedExperiment edgeR:DGEList DESeq:newCountDataset easyRNASeq:knownOrganisms ShortRead:readAligned

Examples

```
## Not run:
library("RnaSeqTutorial")
library(BSgenome.Dmelanogaster.UCSC.dm3)

## creating a count table from 4 bam files
sumExp <- count(filesDirectory=system.file(
                                 "extdata",
                                 package="RnaSeqTutorial"),
                               pattern="[A,C,T,G]{6}\\.bam$",
                               readLength=30L,
                               organism="Dmelanogaster",
                               chr.sizes=seqlengths(Dmelanogaster),
                               annotationMethod="rda",
                               annotationFile=system.file(
                                 "data",
                                 "gAnnot.rda",
                                 package="RnaSeqTutorial"),
                               count="exons"
                               )
    ## the counts
    assays(sumExp)
    ## the sample info
    colData(sumExp)
    ## the features info
    rowData(sumExp)

## End(Not run)
```

DESeq additional methods

Extension for the DESeq package

Description

- multivariateConditions is simply an accessor for the multivariateConditions slot of a CountDataSet object
- plotDispLSD is a function silimar to plotDispEsts that adds a density estimate as a colored heatmap from grey (few) to yellow (many).

Usage

```
## S4 method for signature CountDataSet
multivariateConditions(obj)
```

Arguments

`obj`	An object of class `CountDataSet`

Value

- multivariateConditions returns a boolean describing whether the data to analyze is multivariate or not

Author(s)

Nicolas Delhomme, Bastian Schiffthaler

See Also

`CountDataSet` `plotDispEsts`

Examples

```
## Not run:
## these are helper function for the DESeq package
## refer to its vignette first
cds <- newCountDataSet(countData,conditions)
cds <- estimateSizeFactors(cds)
cds <- estimateDispersions(cds)
mVar <- multivariateConditions(cds)

## End(Not run)
```

`DESeq and edgeR common methods`

DESeq and edgeR common methods

Description

`plotDispersionEstimates(obj,...)` extends the **DESeq** and **edgeR** packages by offering the functionality to plot the dispersion estimate as described in their respective vignettes: `CountDataSet{DESeq}` and `edgeR`.

Usage

```
## S4 method for signature CountDataSet
plotDispersionEstimates(obj, cond = character(1),
  log = "xy", ...)
```

Arguments

`obj`	An object of class `CountDataSet` or of class `DGEList`
`cond`	A character string describing the first condition; for `CountDataSet`{CountDataSet} obj only.
`log`	A character string passed onto `plot.default`; for `CountDataSet`{CountDataSet} obj only.
`...`	Additional plotting parameters; for `CountDataSet`{CountDataSet} obj only.

Details

- `CountDataSet`{DESeq} A character string describing the first condition, to be provided as cond=value
- `edgeR` Unused, just for compatibility.

Value

none

Author(s)

Nicolas Delhomme

Examples

```
## Not run:
## edgeR
## create the object
dgeList <- DGEList(counts,group)
## calculate the size factors
dgeList <- calcNormFactors(dgeList)
## plot them
apply(combn(rownames(dgeList$samples),2),
2,
function(co,obj){plotNormalizationFactors(obj,co[1],co[2])},dgeList)
## the dispersion estimates
plotDispersionEstimates(obj)

## DESeq
## these are helper function for the DESeq package
## refer to its vignette first
cds <- newCountDataSet(countData,conditions)
cds <- estimateSizeFactors(cds)
cds <- estimateDispersions(cds)
plotDispersionEstimates(cds,conditions[1])

## End(Not run)
```

easyRNASeq accessors *Accessors for RNAseq class*

Description

These functions and generics define ‘accessors‘ (to get and set values) for objects in the **easyRNASeq** package.

Usage

```
genomicAnnotation(obj)
readCounts(obj,count=c("exons","features","genes","islands","transcripts"),
summarization=c("bestExons","geneModels"),unique=FALSE)
genomicAnnotation(obj) <- value
```

Arguments

`obj`	An object derived from class `RNAseq`.
`count`	The type of count you want to access, ’genes’,’features’,’exons’,’transcripts’ or ’islands’
`summarization`	If count is set to genes, precise the type of summarization, ’bestExons’ or ’geneModels’
`unique`	For the ’exons’ count only. Should the counts returned be unique for their identifier (i.e. the matrix row names)?
`value`	The replacement value.

Value

Usually, the value of the corresponding slot, or other simple content described on the help page of `easyRNASeq`.

Author(s)

Nicolas Delhomme

Examples

```
rnaSeq<-new("RNAseq")
##set organisme name of an RNAseq object
organismName(rnaSeq) <- "Dmelanogaster"
##get organisme name of an RNAseq object
orgName<-organismName(rnaSeq)
```

easyRNASeq annotation methods

Fetch genic annotation from a gff/gtf file or using biomaRt

Description

The annotation can be retrieved in two ways

- biomaRtUse biomaRt and Ensembl to get organism specific annotation.
- gff/gtfUse a gff or gtf local annotation file.

When using **biomaRt**, it is important that the organismName slot of the RNAseq object is set the prefix of one of the value available using the **biomaRt** listDatasets function, e.g. "Dmelanogaster". When reading from a gff/gtf file, a version 3 formatted gff (gtf are modified gff3 from Ensembl) is expected. The function **genomeIntervals** readGff3 is used to read the data in. Another annotation caveat is the reference names, *i.e.* the chromosome/scaffold names used in the alignment files and those fetched when retrieving the genic annotation might differ. **easyRNASeq** tries to be clever in this case and guess the correspondance. However, it is not always obvious. Organisms were this has been checked can be listed with the *knownOrganisms* function.

Usage

```
## S4 method for signature RNAseq
fetchAnnotation(obj, annotationMethod = c("biomaRt", "gff",
  "gtf"), filename = character(1), ignoreWarnings = FALSE, ...)

## S4 method for signature missing
knownOrganisms()
```

Arguments

obj	An object of class RNAseq
annotationMethod	one of biomaRt, gff, gtf
filename	If the method is gff or gtf, the actual gtf, gff filename
ignoreWarnings	set to TRUE (bad idea! they have a good reason to be there) if you do not want warning messages.
...	See details

Details

... are for additional arguments, passed to the **biomaRt** getBM function or to the readGffGtf internal function that takes an optional arguments: annotation.type that default to "exon". This is used to select the proper rows of the gff or gtf file.

Value

A `RangedData` containing the fetched annotations.

A vector containing the known organisms

Author(s)

Nicolas Delhomme

Nicolas Delhomme

Examples

```
## Not run:
library("RnaSeqTutorial")
obj <- new(RNAseq,
organismName="Dmelanogaster",
readLength=36L,
chrSize=as.list(seqlengths(Dmelanogaster))
)

obj <- fetchAnnotation(obj,
method="gff",
                                    filename=system.file(
"extdata",
"annot.gff",
package="RnaSeqTutorial"))

## End(Not run)
knownOrganisms()
```

`easyRNASeq correction methods`
easyRNASeq count table correction to RPKM

Description

Convert a count table obtained from the easyRNASeq function into an RPKM corrected count table.

Usage

```
## S4 method for signature matrix,ANY,vector,vector
RPKM(obj, from, lib.size = numeric(1),
  feature.size = integer(1), simplify = TRUE, ...)
```

Arguments

`feature.size`	Precise the feature (e.g. exons, genes) sizes. It should be a named numeric list, named after the feature names.
`from`	Determine the kind of coverage to use, choice limited to: exons, features, transcripts, bestExons, geneModels or islands.
`lib.size`	Precise the library size. It should be a named numeric list, i.e. named after the sample names.
`obj`	An object of class `RNAseq` or a `matrix`, see details
`simplify`	If set to TRUE, whenever a feature (exon, feature, ...) is duplicated in the count table, it is only returned once.
`...`	additional arguments. See details

Details

RPKM accepts two sets of arguments:

- RNAseq,character the ... are additional arguments to be passed to the `readCounts` method.
- matrix,named vectornormalize a count matrix by providing the feature sizes (e.g. gene sizes) as a named vector where the names match the row names of the count matrix and the lib sizes as a named vector where the names match the column names of the count matrix.

Value

A `matrix` containing RPKM corrected read counts.

Author(s)

Nicolas Delhomme

See Also

`readCounts`

Examples

```
## Not run:
## get an RNAseq object
rnaSeq <- easyRNASeq(filesDirectory=
     system.file(
"extdata",
package="RnaSeqTutorial"),
pattern="[A,C,T,G]{6}\.bam$",
format="bam",
readLength=36L,
organism="Dmelanogaster",
chr.sizes=as.list(seqlengths(Dmelanogaster)),
annotationMethod="rda",
annotationFile=system.file(
                            "data",
    "gAnnot.rda",
    package="RnaSeqTutorial"),
count="exons",
outputFormat="RNAseq")

## get the RPKM
rpkm <- RPKM(rnaSeq,from="exons")

## the same from a count table
count.table <- readCounts(rnaSeq,count="exons")

## get the RPKM
## verify that the feature are sorted as the count.table
all(.getName(rnaSeq,"exon") == rownames(count.table))
feature.size <- unlist(width(ranges(rnaSeq)))

## verify that the samples are ordered in the same way
all(names(librarySize(rnaSeq)) == colnames(count.table))

## get the RPKM
rpkm <- RPKM(count.table,
feature.size=feature.size,
lib.size=librarySize(rnaSeq))

## End(Not run)
```

---

easyRNASeq coverage methods

Compute the coverage from a Short Read Alignment file

Description

Computes the genomic reads' coverage from a read file in bam format or any format supported by **ShortRead**.

Usage

```
## S4 method for signature RNAseq
fetchCoverage(obj, format = c("aln", "bam"),
  filename = character(1), filter = srFilter(), type = "SolexaExport",
  chr.sel = c(), isUnmappedQuery = FALSE, what = c("rname", "pos",
  "qwidth"), validity.check = TRUE, chr.map = data.frame(),
  ignoreWarnings = FALSE, gapped = TRUE, tag = "NH",
  bp.coverage = FALSE, ...)
```

Arguments

obj	An RNAseq object

`bp.coverage`	a boolean that default to FALSE to decide whether coverage is to be calculated and stored by bp
`chr.map`	A data.frame describing the mapping of original chromosome names towards wished chromosome names. See details.
`chr.sel`	A vector of chromosome names to subset the final results.
`filename`	The full path of the file to use
`filter`	The filter to be applied when loading the data using the "aln" format
`format`	The format of the reads, one of "aln","bam". If not "bam", all the types supported by the ShortRead package are supported too.
`gapped`	Is the bam file provided containing gapped alignments?
`ignoreWarnings`	set to TRUE (bad idea! they have a good reason to be there) if you do not want warning messages.
`isUnmappedQuery`	additional argument for scanBamFlag **Rsamtools**
`tag`	additional argument to the **Rsamtools** scanBamFlag function called internally. The default is NH, to check for multiple mapping.
`type`	The type of data when using the "aln" format. See the **ShortRead** package.
`validity.check`	Shall UCSC chromosome name convention be enforced
`what`	additional argument for ScanBamParam **Rsamtools**
`...`	additional arguments. See details

Details

...for fetchCoverage: Can be used for readAligned method from package **ShortRead** or for scanBamFlag method from package **Rsamtools**.

Value

An `RNAseq` object. The slot readCoverage contains a SimpleRleList object representing a list of coverage vectors, one per chromosome.

Author(s)

Nicolas Delhomme

See Also

`Rle` `ShortRead:readAligned`

Examples

```
## Not run:
library("RnaSeqTutorial")
library(BSgenome.Dmelanogaster.UCSC.dm3)

obj <- new(RNAseq,
organismName="Dmelanogaster",
readLength=36L,
chrSize=as.list(seqlengths(Dmelanogaster))
)

obj <- fetchCoverage(
obj,
format="bam",
                        filename=system.file(
"extdata",
"ACACTG.bam",
                             package="RnaSeqTutorial")
)

## End(Not run)
```

---

easyRNASeq island methods

Identify expressed regions de-novo

Description

Process the coverage to locate regions with a minimum coverage (min.cov). If regions are separated by a gap shorter than a maximum length (max.gap), they are unified. Only islands longer than min.length are returned. These functions are now outdated and would need to be actualized.

Usage

```
## S4 method for signature RNAseq
findIslands(obj, max.gap = integer(1), min.cov = 1L,
  min.length = integer(1), plot = TRUE, ...)
```

Arguments

`obj`	An object of class `RNAseq`
`max.gap`	Maximum gap between two peaks to build an island
`min.cov`	Minimum coverage for an island to be returned
`min.length`	Minimum size of an island to be returned
`plot`	If TRUE, draw plots of coverage distribution. Help the user to select an appropriate value for the minimum coverage.
`...`	See details

Details

... are for providing additional options to the `hist` plot function.

Value

An RNAseq object with the readIsland slot set with a RangedData containing the selected islands and the readCount slot actualized with a list containing the count table per island.

Author(s)

Nicolas Delhomme

Examples

```
## Not run:
## NOTE that this function might need to be actualized
obj <- new(RNAseq,
organismName="Dmelanogaster",
readLength=36L,
chrSize=as.list(seqlengths(Dmelanogaster))
)

obj <- fetchCoverage(
obj,
format="bam",
                        filename=system.file(
"extdata",
"ACACTG.bam",
                             package="RnaSeqTutorial")
)

obj <- findIslands(
obj,
max.gap=10L,
min.cov=10L,
min.length=200L)

## End(Not run)
```

`easyRNASeq package` *Count summarization and normalization pipeline for Next Generation Sequencing data.*

Description

Offers functionalities to summarize read counts per feature of interest, e.g. exons, transcripts, genes, etc. Offers functionalities to normalize the summarized counts using 3rd party packages like DESeq or edgeR.

Details

Package:	easyRNASeq
Type:	Package
Version:	1.8.8
Date:	2014-04-03
License:	Artistic-2.0
LazyLoad:	yes
Depends:	methods, parallel, biomaRt, edgeR, DESeq, genomeIntervals, LSD, Rsamtools, ShortRead, RnaSeqTutorial
Suggests:	BSgenome.Dmelanogaster.UCSC.dm3

Methods

The main function `easyRNASeq` will summarize the counts per feature of interest, for as many samples as provided and will return a count matrix (N*M) where N are the features and M the samples. This data can be corrected to **RPKM** in which case a matrix of corrected value is returned instead, with the same dimensions. Alternatively a `SummarizedExperiment` can be returned and this is expected to be the default in the upcoming version of easyRNASeq (as of 1.5.x). If the necessary sample information are provided, the data can be normalized using either `DESeq` or `edgeR` and the corresponding package object returned. For more insider details, and step by step functions, see:

`ShortRead methods` for pre-processing the data. `easyRNASeq annotation methods` for getting the annotation. `easyRNASeq`

Author(s)

Nicolas Delhomme, Bastian Schiffthaler, Ismael Padioleau

See Also

The class RNAseq specification: `RNAseq`

The default output class specification: `SummarizedExperiment`

The imported packages: `biomaRt` `edgeR` `genomeIntervals` `Biostrings` `BSgenome` `DESeq` `GenomicRanges` `IRanges` `Rsamtools` `ShortRead`

The suggested packages: `parallel` `GenomicFeatures`

Examples

```
## Not run:
library("RnaSeqTutorial")
library(BSgenome.Dmelanogaster.UCSC.dm3)

## creating a count table from 4 bam files
count.table <- easyRNASeq(filesDirectory=
     system.file(
"extdata",
package="RnaSeqTutorial"),
pattern="[A,C,T,G]{6}\.bam$",
format="bam",
readLength=36L,
organism="Dmelanogaster",
chr.sizes=as.list(seqlengths(Dmelanogaster)),
annotationMethod="rda",
annotationFile=system.file(
                            "data",
    "gAnnot.rda",
    package="RnaSeqTutorial"),
count="exons")


## End(Not run)
```

`easyRNASeq summarization methods`
Count methods for RNAseq object

Description

Summarize the read counts per exon, feature, gene, transcript or island.

- `exonCounts`: for that summarization, reads are summarized per exons. An "exon" field is necessary in the annotation object for this to work. See `easyRNASeq annotation methods` for more details on the annotation object.
- `featureCounts` is similar to the 'exons' one. This is just a wrapper to summarize count for genomic features that are not exon related. I.e. one could use it to measure eRNAs. Again, a "feature" field is necessary in the annotation object for this to work.
- `geneCounts` sums the counts per either `bestExons` or `geneModels`. In either case, the annotation object needs to contain both an "exon" and a "gene" field.
- `islandCounts` sums the counts per computed islands.
- `transcriptCounts` sums the counts obtained by exons into their respective transcripts. Note that this often result in counting some reads several times. For this function to work you need both an "exon" and a "transcript" field in your annotation object. To avoid this, one could create transcript specific synthetic exons, i.e. features that would be unique to a transcript. To offer this possibility, transcripts count can be summarized from "features", in which case the annotation object need to have both the "feature" and "transcript" fields defined.

Usage

```
exonCounts(obj)
featureCounts(obj)
transcriptCounts(obj,from="exons")
geneCounts(obj,summarization=c("bestExons","geneModels"),...)
islandCounts(obj,force=FALSE,...)
```

Arguments

`obj`	An object derived from class `RNAseq`,can be a `matrix` for RPKM, see details
`force`	For `islandCount`, force RNAseq to redo `findIsland`
`from`	either "exons" or "features" can be used to summarize per transcript
`summarization`	Method use for summarize genes
`...`	See details

Details

... for

- geneCounts: additional options for the `.geneModelSummarization`
- islandCounts: additional options for `findIslands`

Value

A numeric vector containing count per exon, feature, gene or transcript.

Author(s)

Nicolas Delhomme

See Also

`easyRNASeq annotation methods` `.geneModelSummarization` `findIslands`

Examples

```
## Not run:
## create an RNAseq object
## summarizing 4 bam files by exons
rnaSeq <- easyRNASeq(system.file(
                                 "extdata",
                                 package="RnaSeqTutorial"),
                     organism="Dmelanogaster",
                     chr.sizes=as.list(seqlengths(Dmelanogaster)),
                     readLength=36L,
                     annotationMethod="rda",
                     annotationFile=system.file(
                       "data",
                       "gAnnot.rda",
                       package="RnaSeqTutorial"),
                     format="bam",
                     count="exons",
                     pattern="[A,C,T,G]{6}\.bam$",
                     outputFormat="RNAseq")
## summing up the exons by transcript
rnaSeq <- transcriptCounts(rnaSeq)

## End(Not run)
```

`easyRNASeq,character-method`
easyRNASeq method

Description

This function is a wrapper around the more low level functionalities of the package. Is the easiest way to get a count matrix from a set of read files. It does the following:

- `use ShortRead/Rsamtools methods` for loading/pre-processing the data.
- `fetch the annotations` depending on the provided arguments
- `get the reads coverage` from the provided file(s)
- `summarize the reads` according to the selected summarization features
- `optionally apply` a data correction (i.e. generating RPKM).
- `use edgeR methods` for post-processing the data or
- `use DESeq methods` for post-processing the data (either of them being recommended over RPKM).

Usage

```
## S4 method for signature character
easyRNASeq(filesDirectory = getwd(),
  organism = character(1), chr.sizes = c("auto"), readLength = integer(1),
  annotationMethod = c("biomaRt", "env", "gff", "gtf", "rda"),
  annotationFile = character(1), annotationObject = RangedData(),
  format = c("bam", "aln"), gapped = FALSE, count = c("exons", "features",
  "genes", "islands", "transcripts"), outputFormat = c("matrix",
  "SummarizedExperiment", "DESeq", "edgeR", "RNAseq"), pattern = character(1),
  filenames = character(0), nbCore = 1, filter = srFilter(),
  type = "SolexaExport", chr.sel = c(), summarization = c("bestExons",
  "geneModels"), normalize = FALSE, max.gap = integer(1), min.cov = 1L,
  min.length = integer(1), plot = TRUE, conditions = c(),
  validity.check = TRUE, chr.map = data.frame(), ignoreWarnings = FALSE,
  silent = FALSE, ...)
```

Arguments

`annotationFile`	The location (full path) of the annotation file
`annotationObject`	A `RangedData` or `GRangesList` object containing the annotation.
`annotationMethod`	The method to fetch the annotation, one of "biomaRt","env","gff","gtf" or "rda". All methods but "biomaRt" and "env" require the annotationFile to be set. The "env" method requires the annotationObject to be set.

`chr.map`	A data.frame describing the mapping of original chromosome names towards wished chromosome names. See details.
`chr.sel`	A vector of chromosome names to subset the final results.
`chr.sizes`	A vector or a list containing the chromosomes' size of the selected organism or simply the string "auto". See details.
`conditions`	A vector of descriptor, each sample must have a descriptor if you use outputFormat DESeq or edgeR. The size of this list must be equal to the number of sample. In addition the vector should be named with the filename of the corresponding samples.
`count`	The feature used to summarize the reads. One of 'exons','features','genes','islands' or 'transcripts'. See details.
`filenames`	The name, not the path, of the files to use
`filesDirectory`	The directory where the files to be used are located. Defaults to the current directory.
`filter`	The filter to be applied when loading the data using the "aln" format
`format`	The format of the reads, one of "aln","bam". If not "bam", all the types supported by the **ShortRead** package are supported too. As of version 1.3.5, it defaults to bam.
`gapped`	Is the bam file provided containing gapped alignments?
`ignoreWarnings`	set to TRUE (bad idea! they have a good reason to be there) if you do not want warning messages.
`min.cov`	When computing read islands, the minimal coverage to take into account for calling an island
`min.length`	The minimal size an island should have to be kept
`max.gap`	When computing read islands, the maximal gap size allowed between two islands to merge them
`nbCore`	defines how many CPU core to use when computing the geneModels. Use the default parallel library
`normalize`	A boolean to convert the returned counts in RPKM. Valid when the `outputFormat` is left undefined (i.e. when a matrix is returned) and when it is `DESeq` or `edgeR`. Note that it is not advised to normalize the data prior DESeq or edgeR usage!
`organism`	A character string describing the organism
`outputFormat`	By default, easyRNASeq returns a matrix. If one of `DESeq`,`edgeR`,`RNAseq`, `SummarizedExperiment` is provided then the respective object is returned.
`pattern`	For easyRNASeq, the pattern of file to look for, e.g. "bam$"
`plot`	Whether or not to plot assessment graphs.
`readLength`	The read length in bp
`silent`	set to TRUE if you do not want messages to be printed out.
`summarization`	A character defining which method to use when summarizing reads by genes. So far, only "geneModels" is available.
`type`	The type of data when using the "aln" format. See the ShortRead library.

`validity.check`	Shall UCSC chromosome name convention be enforced? This is only supported for a set of organisms, see `easyRNASeq:knownOrganisms`, otherwise the argument 'chr.map' can be used to complement it.
`...`	additional arguments. See details

Details

- ... Additional arguments for different functions:
  - For the **biomaRt** `getBM` function
  - For the `readGffGtf` internal function that takes an optional arguments: annotation.type that default to "exon" (used to select the proper rows of the gff or gtf file)
  - For the `DESeq` `estimateDispersions` method
  - For to the `list.files` function used to locate the read files.
- the annotationObject When the `annotationMethods` is set to `env` or `rda`, a properly formatted `RangedData` or `GRangesList` object need to be provided. Check the paragraph RangedData in the vignette or the examples at the bottom of this page for examples. The data.frame-like structure of these objects is where `easyRNASeq` will look for the exon, feature, transcript, or gene identifier. Depending on the count method selected, it is essential that the akin column name is present in the annotationObject. E.g. when counting "features", the annotationObject has to contain a "feature" field.
- the chr.map The chr.map argument for the easyRNASeq function only works for an "organismName" of value 'custom' with the "validity.check" parameter set to 'TRUE'. This data.frame should contain two columns named 'from' and 'to'. The row should represent the chromosome name in your original data and the wished name in the output of the function.
- count The count can be summarized by exons, features, genes, islands or transcripts. While exons, genes and transcripts are obvious, "features" describes any features provided by the user, e.g. enhancer loci. These are processed as the exons are. For "islands", it is for an under development function that identifies de-novo expression loci and count the number of reads overlapping them.
- chr.sizes If set to "auto", then the format has to be "bam", in which case the chromosome names and size are extracted from the BAM header

Value

Returns a count table (a matrix of m features x n samples). If the `outputFormat` option has been set, a corresponding object is returned: a `SummarizedExperiment`, a `DESeq:newCountDataset`, a `edgeR:DGEList` or `RNAseq`.

Author(s)

Nicolas Delhomme

See Also

`RNAseq` `SummarizedExperiment` `edgeR:DGEList` `DESeq:newCountDataset` `easyRNASeq:knownOrganisms` `ShortRead:readAligned`

Examples

```
## Not run:
library("RnaSeqTutorial")
library(BSgenome.Dmelanogaster.UCSC.dm3)

## creating a count table from 4 bam files
count.table <- easyRNASeq(filesDirectory=
     system.file(
"extdata",
package="RnaSeqTutorial"),
pattern="[A,C,T,G]{6}\.bam$",
format="bam",
readLength=36L,
organism="Dmelanogaster",
chr.sizes=as.list(seqlengths(Dmelanogaster)),
annotationMethod="rda",
annotationFile=system.file(
                            "data",
    "gAnnot.rda",
    package="RnaSeqTutorial"),
count="exons")

## an example of a chr.map
chr.map <- data.frame(from=c("2L","2R","MT"),to=c("chr2L","chr2R","chrMT"))

## an example of a RangedData annotation
gAnnot <- RangedData(
                     IRanges(
                            start=c(10,30,100),
                            end=c(21,53,123)),
                         space=c("chr01","chr01","chr02"),
                         strand=c("+","+","-"),
                         transcript=c("trA1","trA2","trB"),
                         gene=c("gA","gA","gB"),
                         exon=c("e1","e2","e3"),
                         universe = "Hs19"
                         )

## an example of a GRangesList annotation
grngs <- as(gAnnot,"GRanges")
grngsList<-split(grngs,seqnames(grngs))

## End(Not run)
```

---

edgeR additional methods

Extension for the edgeR package

Description

This method extends the edgeR package by offering the functionality to plot the effect of the normalization factor.

Usage

```
  ## S4 method for signature DGEList,character,character
plotNormalizationFactors(obj = DGEList(),
  cond1 = character(1), cond2 = character(1))
```

Arguments

`obj`	An object of class `DGEList`
`cond1`	A character string describing the first condition
`cond2`	A character string describing the second condition

Value

none

Author(s)

Nicolas Delhomme

Examples

```
## Not run:
## create the object
dgeList <- DGEList(counts,group)
## calculate the sie factors
dgeList <- calcNormFactors(dgeList)
## plot them
apply(combn(rownames(dgeList$samples),2),
2,
function(co,obj){plotNormalizationFactors(obj,co[1],co[2])},dgeList)

## End(Not run)
```

`genomeIntervals additional methods`
Extension for the genomeIntervals package

Description

coerce This method extends the genomeIntervals package by offering the functionality to coerce a `genomeIntervals object` into a `RangedData object` or `GRangesList` object.

type Another way to access the content of the gff type column.

Usage

```
## S4 method for signature Genome_intervals
type(x)
## S4 method for signature Genome_intervals
as(from,to)
```

Arguments

`from`	An object of class `Genome_intervals`
`to`	a character string; either RangedData or GRangesList
`x`	An object of class `Genome_intervals`

Value

coerce A `RangedData` or `GRangesList` containing the result of the coercion.

type The content of the type column, usually a factor or a character vector

Author(s)

Nicolas Delhomme

See Also

`genomeIntervals object` `readGff3 function`

Examples

```
## Not run:
annot<-readGff3(system.file("extdata","annot.gff",package="RnaSeqTutorial")
gAnnot<-as(from=annot,to="RangedData")
type(annot)

## End(Not run)
```

`GenomicRanges additional methods`

Extension of the GenomicRanges package

Description

Return the column name of a `GRanges` or `GRangesList` object.

Usage

```
colnames(x, do.NULL = TRUE, prefix = "col")
```

Arguments

`x`	An object of the `GRanges` or `GRangesList` class
`do.NULL`	see `colnames` for details
`prefix`	see `colnames` for details

Details

It returns the actual column names of the elementMetadata slot of the `GRanges` or `GRangesList` object. The elementMetadata contains a `DataFrame` object used to store additional information provided by the user, such as exon ID in our case.

Value

A vector of column names.

Author(s)

Nicolas Delhomme

See Also

`DataFrame` `GRanges` `GRangesList` `colnames`

Examples

```
## Not run:
## an example of a RangedData annotation
gAnnot <- RangedData(
                     IRanges(
                             start=c(10,30,100),
                             end=c(21,53,123)),
                          space=c("chr01","chr01","chr02"),
                          strand=c("+","+","-"),
                          transcript=c("trA1","trA2","trB"),
                          gene=c("gA","gA","gB"),
                          exon=c("e1","e2","e3"),
                          universe = "Hs19"
                          )

## an example of a GRangesList annotation
grngs <- as(gAnnot,"GRanges")

## accessing the colnames
colnames(grngs)

## creating a GRangesList
grngsList<-split(grngs,seqnames(grngs))

## accessing the colnames
colnames(grngsList)
```

```
## End(Not run)
```

`IRanges additional methods` *Extension of the IRanges package*

Description

Return the ranges of the genomic annotation.

Usage

```
## S4 method for signature RNAseq
ranges(x)
```

Arguments

`x`	An object of the `RNAseq` class

Details

It retrieves the object stored in the genomicAnnotation slot of the RNAseq object and apply the `ranges` function on it. The object retrieved can be of the `RangedData` or `GRangesList` class.

Value

An `IRanges` object.

Author(s)

Nicolas Delhomme

Examples

```
## Not run:
library("RnaSeqTutorial")
obj <- new(RNAseq,
organismName="Dmelanogaster",
readLength=36L,
chrSize=as.list(seqlengths(Dmelanogaster))
)

obj <- fetchAnnotation(obj,
method="gff",
                                filename=system.file(
"extdata",
"annot.gff",
package="RnaSeqTutorial"))
ranges(obj)

## End(Not run)
```

`parallel additional methods`

parallel additional methods

Description

Functions defined in the easyRNASeq package that enhance the parallel package.

Usage

```
## S4 method for signature list,function
parallelize(obj, fun, nnodes = 1, ...)
```

Arguments

`fun`	the function to be applied in parallel
`nnodes`	the number of nodes to use
`obj`	the object which processing has to be parallelizes
`...`	additional arguments passed to the function `fun`

Details

The parallelize function ease the use of the parallel package. If the number of nodes provided by the user is 1, then a simple 'lapply' is used, otherwise a cluster object is created and the object dispatched for parallelization.

Value

the result of the `clusterApply` function.

Author(s)

Nicolas Delhomme

See Also

`clusterApply` `makePSOCKcluster` `stopCluster`

Examples

```
parallelize(list(a<-c(1,2),b<-c(2,1)),sum,nnodes=1)
```

`print methods` *Method to print a RNAseq object*

Description

Print information about a `RNAseq` object.

Usage

```
## S4 method for signature RNAseq
print(x, verbose = FALSE, ...)
```

Arguments

`x`	An object derived from class `RNAseq`
`verbose`	A logical to have a verbose or not output. Default to FALSE
`...`	Additional arguments, currently unused.

Value

Print information about a `RNAseq` object.

Author(s)

Nicolas Delhomme

`RNAseq class` *Class "RNAseq"*

Description

A class holding all the necessary information and annotation to summarize couts (number of reads) per features (i.e. exons or transcripts or genes) for RNA-Seq experiments.

Objects from the Class

Objects can be created by calls of the form `new("RNAseq", ...)`.

Author(s)

Nicolas Delhomme

See Also

- RangedData
- RleList
- easyRNASeq function
- RNAseq accessors
- easyRNASeq annotation methods
- easyRNASeq correction methods
- easyRNASeq coverage methods
- easyRNASeq summarization methods
- print

Examples

```
showClass("RNAseq")
```

ShortRead additional methods *Methods extending the ShortRead package functionalities*

Description

These are functions extending the ShortRead packages capabilities:

Usage

```
demultiplex(obj,barcodes=c(),barcodes.qty=12,barcode.length=6,
edition.dist=2,type=c("independant","within"),index.only=FALSE)
barcodePlot(obj,barcodes=c(),type=c("independant","within"),
barcode.length=6,show.barcode=20,...)
chastityFilter(.name="Illumina Chastity Filter")
naPositionFilter(.name="NA Position Filter")
```

Arguments

`.name`	An internal string describing the filter
`obj`	An object derived from class AlignedRead
`barcodes`	A character vector describing the multiplex (i.e. barcode) sequences used in the experiment.
`barcodes.qty`	An integer describing the number of barcodes
`barcode.length`	An integer describing the barcode length in bp
`edition.dist`	The maximal edition distance (i.e. the number of changes to apply), to accept an incorrectly sequenced barcode.
`index.only`	simply return the index and not the barcode themselves.

`show.barcode`	An integer specifying how many barcodes should be displayed in the final output.
`type`	The type of barcode used. `independent` represents barcodes generated by the illumina protocol; i.e. a separate additional sequencing step performed once the first mate has been sequenced. `within` represents barcodes that are part of the sequenced reads as established by Lefrancois P et al., BMC Genomics, 2009
`...`	additional graphic parameters

Details

- `barcodePlot` Creates a plot showing the barcode distribution of a multiplexed sequencing library.
- `chastityFilter` Creates a `SRFilter` instance that filters SolexaExport read according to the chastity filtering value.
- `demultiplex` Split a single `AlignedRead` object into a list of `AlignedRead` objects according to the barcodes provided by the user.
- `naPositionFilter` Creates a `SRFilter` instance that filters SolexaExport read having an NA position.

When demultiplexing, the function if provided with just the `AlignedRead` will try to find out how many barcodes were used and what they are. This is unwise to do as many barcodes will get wrongly sequenced and not always the most frequent ones are the one you used! It's therefore strongly advised to specify the barcodes' sequences that were used.

Value

- `barcodePlot` returns invisibly the barcode frequencies.
- `chastityFilter` returns a `SRFilter` instance.
- `demultiplex` returns a list of `AlignedRead` objects.
- `naPositionFilter` returns a `SRFilter` instance.

Author(s)

Nicolas Delhomme

See Also

`SRFilter` `AlignedRead`

Examples

```
## Not run:
## the barcode
barcodes=c("ACACTG","ACTAGC","ATGGCT","TTGCGA")

## the multiplexed data
alns <- readAligned(
                    system.file(
                                "extdata",
                                package="RnaSeqTutorial"),
                    pattern="multiplex_export",
                    filter=compose(
                      chastityFilter(),
                      nFilter(2),
                      chromosomeFilter(regex="chr")),
                    type="SolexaExport",
                    withAll=TRUE)

## barcode plot
barcodePlot(alns,
            barcodes=barcodes,
            type="within",
            barcode.length=6,
            show.barcode=20,
            main="All samples",
            xlim=c(0,0.5))

## demultiplexing
dem.alns <- demultiplex(alns,
                        barcodes=barcodes,
                        edition.dist=2,
                        barcodes.qty=4,
                        type="within")

## plotting again
par(mfrow=c(2,2))
barcode.frequencies <- lapply(
                              names(dem.alns$barcodes),
                              function(barcode,alns){
                                barcodePlot(
                                            alns$barcodes[[barcode]],
                                            barcodes=barcode,
                                            type="within",barcode.length=6,
                                            show.barcode=20,
                                            main=paste(
                                              "Expected barcode:",
                                              barcode))
                              },dem.alns)

## End(Not run)
```

`show methods` *Display the content of a RNAseq object*

Description

Display the content of a `RNAseq` object.

Usage

```
## S4 method for signature RNAseq
show(object)
```

Arguments

`object`	Any R object

Methods

list("signature(object = \"RNAseq\")") Display the values of the different slots of the `RNAseq` object.

Index