Package 'easyRNASeq'

September 24, 2012

Version 1.2.5

Date 2011-08-24

Type Package

Title Count summarization and normalization for RNA-Seq data.

Author Nicolas Delhomme, Ismael Padioleau

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 graphics, methods, parallel, utils, genomeIntervals (>= 1.12.0), Biobase (>= 2.16.0), BiocGenerics (>= 0.2.0), biomaRt (>= 2.12.0), edgeR (>= 2.6.10), Biostrings (>= 2.24.1),BSgenome (>= 1.24.0), DESeq (>= 1.8.3), GenomicRanges (>= 1.8.12), IRanges (>= 1.14.4), Rsamtools (>= 1.8.6), ShortRead (>= 1.14.4)

Suggests BSgenome.Dmelanogaster.UCSC.dm3 (>= 1.3.17), GenomicFeatures (>= 1.8.2), RnaSeqTutorial (>= 0.0.8)

License Artistic-2.0

LazyLoad yes

Enhances edgeR, genomeIntervals, DESeq, ShortRead

biocViews GeneExpression, RNAseq, Genetics, Preprocessing

R topics documented:

`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:	1.2.5
Date:	2012-08-24
License:	Artistic-2.0
LazyLoad:	yes
Depends:	methods, parallel, biomaRt, edgeR, DESeq, genomeIntervals, Rsamtools, ShortRead, RnaSeqTutorial
Suggests:	BSgenome.Dmelanogaster.UCSC.dm3

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 will be 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. 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. `easyRNA`

Author(s)

Nicolas Delhomme Maintainer: Nicolas Delhomme <delhomme@embl.de>

See Also

The class RNAseq specification: `RNAseq`

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)
```

`accessors` *Accessors for RNAseq class*

Description

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

Usage

```
##chrSize
chrSize(obj)<-value
chrSize(obj)

##fileName
fileName(obj)<-value
fileName(obj)

##geneModel
geneModel(obj)<-value
geneModel(obj)
```

```
##genomicAnnotation
genomicAnnotation(obj)<-value
genomicAnnotation(obj)

##librarySize
librarySize(obj)<-value
librarySize(obj)

##organismName
organismName(obj)<-value
organismName(obj)

##readCounts
readCounts(obj)<-value
readCounts(obj,count=c("exons","features","genes","islands","transcripts"),summarization=c("best

##readCoverage
readCoverage(obj)<-value
readCoverage(obj)

##readIslands
readIslands(obj)<-value
readIslands(obj)

##readLength
readLength(obj)<-value
readLength(obj)
```

Arguments

obj	An object derived from class RNAseq. The default is to extract or set the contents of a slot of the corresponding name.
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)
```

DESeq 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

```
plotDispersionEstimates(obj,...)
```

Arguments

`obj`	An object of class `CountDataSet` or of class `DGEList`
`...`	See details

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 sie 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)
```

DESeq methods *Extension for the DESeq package*

Description

- `multivariateConditions` is simply an accessor for the `multivariateConditions` slot of a `CountDataSet` object.

Usage

```
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

See Also

`CountDataSet`

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)
```

easyRNASeq *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

```
easyRNASeq(filesDirectory=character(1),
           organism=character(1),
                chr.sizes=c(),
           readLength=integer(1),
           annotationMethod=c("biomaRt","env","gff","gtf","rda"),
           annotationFile=character(1),
   annotationObject=RangedData(),
           format=c("aln","bam"),gapped = FALSE,
           count=c('exons','features','genes','islands','transcripts'),
           outputFormat=c("DESeq","edgeR","matrix","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
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','genes','islands' or 'transcripts'
filenames	The name, not the path, of the files to use
filesDirectory	The directory where the files to be used are located
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.
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 count matrix. If one of DESeq,edgeR,RNAseq is provided then the respective object will be 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
...	additional arguments. See details

Details

- ... for the easyRNASeq call 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" (used to select the proper rows of the gff or gtf file) or to the `DESeq estimateDispersions` method.

- 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.

Value

Returns a count table (a matrix of m features x n samples) unless the outputFormat option has been set, in which case an object of type DESeq:newCountDataset or edgeR:DGEList or RNAseq is returned

Author(s)

Nicolas Delhomme

See Also

RNAseq edgeR:DGEList DESeq:newCountDataset

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)
```

`easyRNASeq annotation methods`

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

Description

The annotation can be retrieved in two ways

- biomaRt Use biomaRt and Ensembl to get organism specific annotation.
- gff/gtf Use 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.

Usage

```
fetchAnnotation(obj,method=c("biomaRt","gff","gtf"),filename=character(1),ignoreWarnings=FALSE
```

Arguments

`obj`	An object of class `RNAseq`
`method`	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.

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"))

## End(Not run)
```

`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

```
RPKM(obj,
from=c("exons","features","transcripts","bestExons","geneModels","islands"),
lib.size=numeric(1),
feature.size=numeric(1),
unique=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

unique	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 vector normalize 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

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

Usage

```
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,...)
```

Arguments

`obj`	An RNAseq object
`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**
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 de-novo annotation 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

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

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)
```

---

edgeR 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

```
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 methods

Extension for the genomeIntervals package

Description

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

Arguments

from	An object of class Genome_intervals

Value

A RangedData containing the result of the coercion.

Author(s)

Nicolas Delhomme

See Also

genomeIntervals object readGff3 function

Examples

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

## End(Not run)
```

GenomicRanges methods *Extension of the GenomicRanges package*

Description

Return the column name of a GRanges or GRangesList object.

Arguments

obj	An object of the GRanges or GRangesList class

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

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 methods *Extension of the IRanges package*

Description

Return the ranges of the genomic annotation.

Arguments

obj	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)
```

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

Description

Print information about a `RNAseq` object.

Arguments

`rnaSeq`	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", ...)`.

Slots

`chrSize`: A named `"list"` containing the chromosomes' size

`fileName`: Name of the last processed file

`geneModel`: A `RangedData` containing gene models

`genomicAnnotation`: Object of class `RangedData` or `GRangesList` containing genomic annotations, i.e. exon coordinates, xrefs, *etc.*

`librarySize`: A named numeric list describing the library size of every sample.

`organismName`: A `"character"` describing the organism from which the sample was derived.

`readCounts`: A `"list"` of `"list"` containing summarized counts per summarization features, i.e. per exon, or transcript or gene

`readCoverage`: A `RleList` containing the provided sample coverage

`readIslands`: A `RangedData` containing read islands

`readLength`: An `"integer"` specifying the read length

Methods

chrSize signature(obj = "RNAseq"): see RNAseq accessors
chrSize<- signature(obj = "RNAseq"): see RNAseq accessors
easyRNASeq signature(normalizeMethod = "RNAseq"): see easyRNASeq function
exonCounts signature(obj = "RNAseq"): easyRNASeq summarization methods
fetchAnnotation signature(obj = "RNAseq"): see easyRNASeq annotation methods
fetchCoverage signature(obj = "RNAseq"): see easyRNASeq methods
fileName signature(obj = "RNAseq"): see RNAseq accessors
fileName<- signature(obj = "RNAseq"): see RNAseq accessors
geneCounts signature(obj = "RNAseq"): easyRNASeq summarization methods
geneModel signature(obj = "RNAseq"): see RNAseq accessors
geneModel<- signature(obj = "RNAseq"): see RNAseq accessors
genomicAnnotation signature(obj = "RNAseq"): see RNAseq accessors
genomicAnnotation<- signature(obj = "RNAseq"): see RNAseq accessors
islandCounts signature(obj = "RNAseq"):easyRNASeq summarization methods
organismName signature(obj = "RNAseq"): see RNAseq accessors
organismName<- signature(obj = "RNAseq"): see RNAseq accessors
print signature(x = "RNAseq"): see print
readCounts signature(obj = "RNAseq"): see RNAseq accessors
readCounts<- signature(obj = "RNAseq"): see RNAseq accessors
readCoverage signature(obj = "RNAseq"): see RNAseq accessors
readCoverage<- signature(obj = "RNAseq"): see RNAseq accessors
readLength signature(obj = "RNAseq"): see RNAseq accessors
readLength<- signature(obj = "RNAseq"): see RNAseq accessors
RPKM signature(obj = "RNAseq"): easyRNASeq correction methods
show signature(object = "RNAseq"): see show
transcriptCounts signature(obj = "RNAseq"): easyRNASeq summarization methods

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-methods *Methods extending the ShortRead package functionalities*

Description

These are functions extending the ShortRead packages capabilities:

- 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.

Usage

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

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.

Methods

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

Index