

# Package ‘easyRNASeq’

October 16, 2018

**Version** 2.16.0

**Date** 2018-02-23

**Type** Package

**Title** Count summarization and normalization for RNA-Seq data

**Author** Nicolas Delhomme, Ismael Padioleau, Bastian Schiffthaler, Niklas Maehler

**Maintainer** Nicolas Delhomme <nicolas.delhomme@umu.se>

**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.

**Imports** Biobase (>= 2.39.1), BiocGenerics (>= 0.25.1), BiocParallel (>= 1.13.1), biomaRt (>= 2.35.8), Biostrings (>= 2.47.6), DESeq (>= 1.31.0), edgeR (>= 3.21.6), GenomeInfoDb (>= 1.15.2), genomeIntervals (>= 1.35.1), GenomicAlignments (>= 1.15.6), GenomicRanges (>= 1.31.8), SummarizedExperiment (>= 1.9.9), graphics, IRanges (>= 2.13.13), LSD (>= 3.0), locfit, methods, parallel, Rsamtools (>= 1.31.2), S4Vectors (>= 0.17.25), ShortRead (>= 1.37.1), utils

**Suggests** BiocStyle (>= 2.7.8), BSgenome (>= 1.39.0), BSgenome.Dmelanogaster.UCSC.dm3 (>= 1.4.0), curl, knitr, rmarkdown, RUnit (>= 0.4.31)

**License** Artistic-2.0

**LazyLoad** yes

**VignetteBuilder** knitr

**biocViews** GeneExpression, RNASeq, Genetics, Preprocessing

**RoxygenNote** 6.0.1

**git\_url** <https://git.bioconductor.org/packages/easyRNASeq>

**git\_branch** RELEASE\_3\_7

**git\_last\_commit** 09fe48d

**git\_last\_commit\_date** 2018-04-30

**Date/Publication** 2018-10-15

**R topics documented:**

AnnotParam class . . . . .	2
BamParam class . . . . .	3
basename methods . . . . .	4
createSyntheticTranscripts,AnnotParamCharacter-method . . . . .	5
Defunct functions . . . . .	6
DESeq additional methods . . . . .	7
easyRNASeq accessors . . . . .	8
easyRNASeq annotation methods . . . . .	9
easyRNASeq AnnotParam accessors . . . . .	10
easyRNASeq AnnotParam constructor . . . . .	11
easyRNASeq BamParam accessors . . . . .	12
easyRNASeq BamParam constructor . . . . .	13
easyRNASeq correction methods . . . . .	14
easyRNASeq coverage methods . . . . .	16
easyRNASeq defunct annotation methods . . . . .	17
easyRNASeq GenomicRanges package extension . . . . .	18
easyRNASeq island methods . . . . .	19
easyRNASeq package . . . . .	21
easyRNASeq RnaSeqParam accessors . . . . .	22
easyRNASeq RnaSeqParam constructor . . . . .	23
easyRNASeq summarization methods . . . . .	24
easyRNASeq,character-method . . . . .	26
easyRNASeq-datasets . . . . .	30
edgeR additional methods . . . . .	30
file.exists methods . . . . .	31
genomeIntervals additional methods . . . . .	32
getBamFileList . . . . .	33
IRanges additional methods . . . . .	34
parallel additional methods . . . . .	35
print methods . . . . .	36
RNAseq class . . . . .	36
RnaSeqParam class . . . . .	37
ShortRead additional methods . . . . .	38
show methods . . . . .	40
simpleRNASeq,BamFileList,RnaSeqParam-method . . . . .	41
validate,BamFile-method . . . . .	42
<b>Index</b>	<b>44</b>

---

AnnotParam class	<i>Class "AnnotParam"</i>
------------------	---------------------------

---

**Description**

A class holding all the necessary parameters to retrieve the necessary annotation for processing an RNA-Seq experiment.

### Objects from the Class

Objects can be created by calls of the form `new("AnnotParamCharacter", ...)` or `new("AnnotParamObject", ...)` (both subject to API changes) or using the `AnnotParam` constructor (failsafe, preferred). The class `AnnotParam` in itself is virtual and hence cannot be instantiated.

### Author(s)

Nicolas Delhomme

### See Also

- [RnaSeqParam](#)
- [RnaSeqParam constructor](#)
- [RnaSeqParam accessors](#)
- [simpleRNASeq function](#)
- [AnnotParam constructor](#)

### Examples

```
showClass("AnnotParam")
```

---

BamParam class

*Class "BamParam"*

---

### Description

A class describing the parameters of a bam file issued from an RNA-Seq experiment.

### Objects from the Class

Objects can be created by calls of the form `new("BamParam", ...)` or using the `BamParam` constructor.

### Slots from the Class

The `BamParam` class has the following slots:

- `paired`
- `stranded`
- `strandProtocol`
- `yieldSize`

all of which can be accessed using the accordingly names accessor.

### Author(s)

Nicolas Delhomme

**See Also**

- [BamParam](#) accessors
- [RnaSeqParam](#)
- [RnaSeqParam](#) constructor
- [RnaSeqParam](#) accessors
- [simpleRNASeq](#) function
- [AnnotParam](#)
- [AnnotParam](#) constructor

**Examples**

```
showClass("BamParam")
```

---

basename methods	<i>Extend the basename function to display Rsamtools BamFile class basename</i>
------------------	---

---

**Description**

Display the basename of the bam file represented by a [BamFile](#) object.

**Usage**

```
## S4 method for signature 'BamFile'  
basename(path)
```

**Arguments**

path            an object of class [BamFile](#) or [BamFileList](#)

**Methods**

**list("signature(object = \"BamFile\")")** Display the basename of the bam file linked to by a [BamFile](#) object.

---

createSyntheticTranscripts,AnnotParamCharacter-method  
*Methods to create synthetic transcripts*

---

## Description

This function create a set of synthetic transcripts from a provided annotation file in "gff3" or "gtf" format. As detailed in <http://www.epigenesys.eu/en/protocols/bio-informatics/1283-guidelines-for-rna-seq-data-analysis>, one major caveat of estimating gene expression using aligned RNA-Seq reads is that a single read, which originated from a single mRNA molecule, might sometimes align to several features (e.g. transcripts or genes) with alignments of equivalent quality. This, for example, might happen as a result of gene duplication and the presence of repetitive or common domains. To avoid counting unique mRNA fragments multiple times, the stringent approach is to keep only uniquely mapping reads - being aware of potential consequences. Not only can "multiple counting" arise from a biological reason, but also from technical artifacts, introduced mostly by poorly formatted gff3/gtf annotation files. To avoid this, it is best practice to adopt a conservative approach by collapsing all existing transcripts of a single gene locus into a "synthetic" transcript containing every exon of that gene. In the case of overlapping exons, the longest genomic interval is kept, i.e. an artificial exon is created. This process results in a flattened transcript - a gene structure with a one (gene) to one (transcript) relationship.

## Usage

```
## S4 method for signature 'AnnotParamCharacter'
createSyntheticTranscripts(obj,
  features = c("mRNA", "miRNA", "tRNA", "transcript"), verbose = TRUE)

## S4 method for signature 'character'
createSyntheticTranscripts(obj, features = c("mRNA",
  "miRNA", "tRNA", "transcript"), verbose = TRUE,
  output = c("Genome_intervals", "GRanges"), input = c("gff3", "gtf"))
```

## Arguments

obj	a <a href="#">AnnotParamCharacter</a> object or the annotation filename as a character string
features	one or more of 'mRNA', 'miRNA', 'tRNA', 'transcript'
verbose	increase the verbosity (default TRUE)
output	the output type, one of 'Genome_intervals' or 'GRanges'
input	the type of input, one of 'gff3' or 'gtf'
...	If obj is a character string, input and output - see below

## Details

The createSyntheticTranscripts function implements this, taking advantage of the hierarchical structure of the gff3/gtf file. Exon features are related to their transcript (parent), which themselves derives from their gene parents. Using this relationship, exons are combined per gene into a flattened transcript structure. Note that this might not avoid multiple counting if genes overlap on opposing strands. There, only strand specific sequencing data has the power to disentangle these situations.

As gff3/gtf file can contain a large number of feature types, the `createSyntheticTranscripts` currently only supports: *mRNA*, *miRNA*, *tRNA* and *transcript*. Please contact me if you need additional features to be considered. Note however, that I will only add features that are part of the [sequenceontology.org](http://sequenceontology.org) SOFA (SO\_Feature\_Annotation) ontology.

### Value

Depending on the obj class.

- `AnnotParamCharacter`: a `AnnotParamObject` object
- a character filename: depending on the selected output value, a `Genome_intervals` or a `GRanges` object.

### Author(s)

Nicolas Delhomme

### See Also

- For the input:
  - `AnnotParam`
- For the output:
  - `AnnotParam`
  - `Genome_intervals`
  - `GRanges`

### Examples

```
# get the example file
library(curl)
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
                               "master/tutorial/easyRNASeq/",
                               "Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"),
                       "Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"))
# create the AnnotParam
annotParam <- AnnotParam(
  datasource="Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz",
  type="gtf")

# create the synthetic transcripts
annotParam <- createSyntheticTranscripts(annotParam, verbose=FALSE)
```

---

Defunct functions

*The following function are defunct:*

- `easyRNASeq`
  - `fetchCoverage`
  - `fetchAnnotation`
  - `knownOrganisms`
  - `plotDispersionEstimates, DGEList-method`
-

**Description**

- The `plotDispersionEstimates`, `DGEList`-method function is superseded by the `plotBCV` function as the **edgeR** `DGEList` object structure changed
- The `easyRNASeq` function is superseded by the `simpleRNASeq` function to consolidate and prune the overall package. The changes are based on user comments and on the general standardization occurring in the field.
- The `fetchCoverage` function only had two parameters deprecated as the consequence of the package consolidation. As the `scanBam` function is not called directly anymore but through higher level functions (from the `GenomicRanges` package), the 'what' and 'isUnmapped-Query' parameters were obsolete.

---

 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 similar to `plotDispEsts` that adds a density estimate as a colored heatmap from grey (few) to yellow (many).
- `plotDispersionEstimates` offers the functionality to plot the dispersion estimate as described in the **DESeq** vignette.

**Usage**

```

multivariateConditions(obj)
plotDispLSD(obj, name = NULL, ymin,
linecol = "#00000080", xlab = "mean of normalized counts",
ylab = "dispersion", log = "xy", cex = 0.45, ...)
plotDispersionEstimates(obj, cond, log, ...)

```

**Arguments**

<code>obj</code>	An object of class <code>CountDataSet</code> .
<code>cex</code>	The standard <code>plot.default</code> parameter.
<code>cond</code>	A character string describing the first condition.
<code>linecol</code>	Defines the line color.
<code>log</code>	A character string passed onto <code>plot.default</code> .
<code>name</code>	Argument passed to the <b>DESeq</b> <code>fitInfo</code> function.
<code>xlab</code>	The standard <code>plot.default</code> parameter.
<code>ylab</code>	The standard <code>plot.default</code> parameter.
<code>ymin</code>	A numeric value defining the lower limit for the y axis.
<code>...</code>	Additional plotting parameters.

**Value**

- `multivariateConditions` returns a boolean describing whether the data to analyze is multivariate or not
- `plotDispLSD` and `plotDispersionEstimates` returns nothing

**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)
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**

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

```
# This class is deprecated and as such there are no exmples of its use
```

---

easyRNASeq annotation methods

*Get genic annotation from a gff3/gtf file or using biomaRt*

---

**Description**

The annotation can be retrieved in two ways

- `biomaRtUse` `biomaRt` and `Ensembl` to get organism specific annotation.
- `gff3/gtfUse` a `gff3` or `gtf` local annotation file.
- When using **biomaRt**, it is important that the `organism` argument to `AnnotParam` is set the prefix of one of the value available using the **biomaRt** `listDatasets` function, e.g. "Dmelanogaster".
- When reading from a `gff3/gtf` file, a version 3 formatted `gff` or a `gtf` (an `Ensembl` defined `gff2` version) is expected. The function **genomeIntervals** `readGff3` is used to import the data.

**Usage**

```
## S4 method for signature 'AnnotParam'  
getAnnotation(obj, verbose = FALSE, ...)
```

**Arguments**

<code>obj</code>	An object of class <code>AnnotParam</code>
<code>verbose</code>	a boolean to turn on verbosity
<code>...</code>	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 `GRanges` containing the fetched annotations.

**Author(s)**

Nicolas Delhomme

**Examples**

```
## Not run:
library("RnaSeqTutorial")
getAnnotation(
  AnnotParam(
    organism="Dmelanogaster",
    datasource=system.file(
      "extdata",
      "Dmel-mRNA-exon-r5.52.gff3",
      package="RnaSeqTutorial"),
    type="gff3"
  ))

## End(Not run)
```

---

easyRNASeq AnnotParam accessors

*Accessors for AnnotParam class*

---

**Description**

These functions and generics define ‘accessors’ (to get and set values) for [AnnotParam](#) objects within the **easyRNASeq** package. Implemented are:

- `datasource`
- `type`

**Usage**

```
datasource(object)
## S4 method for signature 'AnnotParam'
type(x)
```

**Arguments**

<code>object</code>	An object derived from class <code>AnnotParam</code> .
<code>x</code>	An object derived from class <code>AnnotParam</code> .

**Value**

The value of the corresponding slot.

**Author(s)**

Nicolas Delhomme

**See Also**

The [AnnotParam](#) class. The type and organism generics are imported from the [BSgenome](#) and [Biostrings](#) package, respectively.

**Examples**

```
library(curl)
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
                              "master/tutorial/easyRNASeq/Dmel-mRNA-exon-r5.52.gff3.gz"),
                      "Dmel-mRNA-exon-r5.52.gff3.gz"))

annot <- AnnotParam(datasource="Dmel-mRNA-exon-r5.52.gff3.gz")
# get the datasource Parameter
datasource(annot)
```

---

easyRNASeq AnnotParam constructor

*AnnotParam constructor*

---

**Description**

This constructs a [AnnotParam](#) object. The datasource parameter (see details) is mandatory, however other parameters, *i.e.* when the datasource is not a [GRanges](#) or [RangedData](#) default to "genes" and gff3", indicating that the datasource is in the gff3 format and that the contained information needs to be grouped by "genes". This representing the most common use case. Hence, it is left to the user to refine the parameters accordingly to the annotation he is providing or wishes to retrieve.

**Usage**

```
## S4 method for signature 'character'
AnnotParam(datasource = character(0), type = c("gff3",
      "biomaRt", "gtf", "rda"))
```

**Arguments**

datasource      a character or a [RangedData](#) or a [GRanges](#) object. See details.  
 type            one of NULL, biomaRt, gff3, gtf or rda. Default to NULL. See details.

**Details**

Note that calling the constructor without argument fails, as the datasource is a mandatory parameter. Calling the constructor with additional (not all) parameters will affect the value of the selected parameters, leaving the other parameters unaffected. There are three parameters for an [AnnotParam](#) object:

- datasourceIf no type is provided, the datasource should be either a [GRanges](#)(preferred) or a [RangedData](#) (subject to future deprecation) object containing the genic information. These can be obtained using the [getAnnotation](#) function.
- typeOne of biomaRt, gff3, gtf or rda. The default is "gff3". In all cases, the datasource is a character describing:

- For biomaRt, the name of the organism as known by the ensembl Mart, *e.g.* dmelanogaster or hsapiens.
- For gff3, gtf or rda, the filename (including the full or relative path).

### See Also

- [GRanges](#)
- [RangedData](#)
- [getAnnotation](#)

### Examples

```
# create an object to retrieve annotation from biomaRt
annotParam <- AnnotParam(datasource="Hsapiens",type="biomaRt")

# get the datasource and type
datasource(annotParam)
type(annotParam)

# create an object to retrieve annotation from an rda object
library(curl)
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
                               "master/tutorial/easyRNASeq/gAnnot.rda"), "gAnnot.rda"))
annotParam <- AnnotParam(datasource="gAnnot.rda",type="rda")
```

---

easyRNASeq BamParam accessors

*Accessors for BamParam class*

---

### Description

These functions and generics define ‘accessors’ (to get and set values) for [BamParam](#) objects within the **easyRNASeq** package.

### Usage

```
yieldSize(object,...)
paired(object)
stranded(object)
strandProtocol(object)
```

### Arguments

object	An object derived from class BamParam.
...	Additional parameter inherited from the <a href="#">Rsamtools</a> package <a href="#">yieldSize</a> function. Ignored here.

### Value

The value of the corresponding slot.

**Author(s)**

Nicolas Delhomme

**See Also**The [BamParam](#) class The [RnaSeqParam](#) [yieldSize](#) accessor**Examples**

```
bp <- BamParam()
## get the yieldSize Parameter
ysize <-yieldSize(bp)
```

---

 easyRNASeq BamParam constructor

*BamParam constructor*


---

**Description**

This constructs a [BamParam](#) object. The default parameters are derived from the currently most common RNA-Seq experimental use-case and are detailed below:

- paired is TRUE, *i.e.* paired-end sequencing is expected.
- stranded is FALSE *i.e.* stranded sequencing is not expected.
- yieldSize is set to 1,000,000. This is the amount of reads iteratively processed from the bam file stream. It is a compromise between speed, process-parallelization and memory usage.

**Usage**

```
## S4 method for signature 'ANY'
BamParam(paired = TRUE, stranded = FALSE,
  strandProtocol = c("reverse", "forward"), yieldSize = 1000000L)
```

**Arguments**

paired	boolean whether the BAM file contains paired-end data or not
stranded	boolean whether the reads are strand specific
strandProtocol	factor with values 'reverse' and 'forward' specifying the type of strand specificity protocol. 'reverse', the reads are on the opposite strand to the gene; typical for Illumina TRUSEQ strand-specific protocol.
yieldSize	the amount of reads to be streamed at a time. Default to 1M

**Details**

Calling the constructor without argument result in the default parameter described above to be returned. Calling the constructor with any parameter will affect the value of the selected parameters, leaving the other parameters unaffected.

**Examples**

```
# the defaults
BamParam()

# change the default
BamParam(paired=FALSE)
BamParam(stranded=TRUE,yieldSize=1L)
BamParam(stranded=TRUE,strandProtocol="forward",yieldSize=1L)
```

---

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 = c("exons", "features",
  "transcripts", "bestExons", "geneModels", "islands"), lib.size = numeric(1),
  feature.size = integer(1), simplify = TRUE, ...)
```

**Arguments**

obj	An object of class <a href="#">RNAseq</a> or a matrix, see details
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.
feature.size	Precise the feature (e.g. exons, genes) sizes. It should be a named numeric list, named after the feature names.
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(), validity.check = TRUE, chr.map = data.frame(),
  ignoreWarnings = FALSE, gapped = TRUE, paired = FALSE,
  stranded = FALSE, bp.coverage = FALSE, ...)
```

## Arguments

obj	An <a href="#">RNAseq</a> object
format	The format of the reads, one of "aln", "bam". If not "bam", all the types supported by the ShortRead package are supported too.
filename	The full path of the file to use
filter	The filter to be applied when loading the data using the "aln" format
type	The type of data when using the "aln" format. See the <b>ShortRead</b> package.
chr.sel	A vector of chromosome names to subset the final results.
validity.check	Shall UCSC chromosome name convention be enforced
chr.map	A data.frame describing the mapping of original chromosome names towards wished chromosome names. See details.
ignoreWarnings	set to TRUE (bad idea! they have a good reason to be there) if you do not want warning messages.
gapped	Is the bam file provided containing gapped alignments?
paired	Is the bam file containing PE reads?
stranded	Is the bam file from a strand specific protocol?
bp.coverage	a boolean that default to FALSE to decide whether coverage is to be calculated and stored by bp
...	additional arguments. See details

## Details

... for fetchCoverage: Can be used for readAligned method from package **ShortRead**. The use of the dots for the scanBamFlag method from package **Rsamtools** has been deprecated, as were the 'what' and 'isUnmappedQuery' argument to the function

## 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 defunct annotation methods  
*Defunct annotation function*

---

**Description**

The `fetchAnnotation` and `knownOrganisms` function are now defunct. The `fetchAnnotation` function has been replaced by the [getAnnotation](#) method.

**Author(s)**

Nicolas Delhomme

---

easyRNASeq GenomicRanges package extension

*Extension of the GenomicRanges package*

---

## Description

Describes extensions to the [GenomicRanges](#) package. For [GRanges](#) and [GRangesList](#) objects:

- `colnames` returns the column name of a [GRanges](#) or [GRangesList](#) object.
- `unsafeAppend` appends two [GAlignments](#) object together bypassing most sanity checks. Faster than the standard `c` or `append` function.

## Usage

```
colnames(x, do.NULL = TRUE, prefix = "col")
unsafeAppend(obj1, obj2)
```

## Arguments

<code>x</code>	An object of the <a href="#">GRanges</a> or <a href="#">GRangesList</a> class
<code>do.NULL</code>	see <a href="#">colnames</a> for details
<code>prefix</code>	see <a href="#">colnames</a> for details
<code>obj1</code>	A <a href="#">GAlignments</a> object
<code>obj2</code>	A <a href="#">GAlignments</a> object

## Details

- `colnames` 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.
- `unsafeAppend` appends two [GAlignments](#) objects.

## Value

- `colnames`: A vector of column names.
- `unsafeAppend`: A [GAlignments](#) object

## Author(s)

Nicolas Delhomme

## See Also

- [DataFrame](#)
- [GRanges](#)
- [GRangesList](#)
- [GAlignments colnames](#)

**Examples**

```

# 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("+","+","-"),
  transcripts=c("trA1","trA2","trB"),
  gene=c("gA","gA","gB"),
  exon=c("e1","e2","e3")
)

# 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)

# For unsafeAppend
library(GenomicAlignments)
unsafeAppend(GAlignments(),GAlignments())

```

---

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))
)

library(curl)
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

obj <- fetchCoverage(obj, format="bam", filename="ACACTG.bam")

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

## End(Not run)
```

---

easyRNASeq package	<i>Count summarization and normalization pipeline for Next Generation Sequencing data.</i>
--------------------	--

---

## 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](#).

## 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 [RangedSummarizedExperiment](#) 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 Schifftaler, Ismael Padioleau

## See Also

The class RNAseq specification: [RNAseq](#)

The default output class specification: [RangedSummarizedExperiment](#)

The imported packages: [biomaRt](#) [BiocParallel](#) [edgeR](#) [genomeIntervals](#) [Biostrings](#) [BSgenome](#) [DESeq](#) [GenomicRanges](#) [IRanges](#) [Rsamtools](#) [ShortRead](#)

The suggested packages: [parallel](#) [GenomicFeatures](#)

The following classes and functions that are made available from other packages:

- Classes [BamFileList](#) [CountDataSet](#) [RangedData](#) [RangedSummarizedExperiment](#)
- Functions/Methods [DESeq](#) [estimate size factor](#) and [estimate dispersion functions](#)  
[The RangedSummarizedExperiment assay accessor](#) [The locfit function](#)  
[The BamFileList constructor](#) [The IRanges constructor](#) [The RangedData constructor](#)  
[For the SRFilterResult,](#) [chromosomeFilter](#), [compose](#) and [nFilter](#) methods

## Examples

```
# get the example annotation file - we retrieve a gtf file from GitHub
library(curl)
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
"master/tutorial/easyRNASeq/Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"),
"Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"))
```

```

# get the example data files - we retrieve a set of example bam files
# from Github using curl, as well as their index.
invisible(sapply(c("ACACTG","ACTAGC"),function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/",bam,".bam"),paste0(bam,".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/",bam,".bam.bai"),paste0(bam,".bam.bai"))
}))

# create the AnnotParam
annotParam <- AnnotParam(
  datasource="Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz",
  type="gtf")

# create the synthetic transcripts
annotParam <- createSyntheticTranscripts(annotParam,verbose=FALSE)

# create the RnaSeqParam
rnaSeqParam <- RnaSeqParam(annotParam=annotParam,countBy="gene")

# get the bamfiles
bamFiles <- getBamFileList(dir(pattern="^[A,T].*\\.bam$",full.names=TRUE))

# get a RangedSummarizedExperiment containing the counts table
sexp <- simpleRNASeq(
  bamFiles=bamFiles,
  param=rnaSeqParam,
  verbose=TRUE
)

# get the counts
assays(sexp)$genes

```

---

easyRNASeq RnaSeqParam accessors

*Accessors for RnaSeqParam class*

---

## Description

These functions and generics define ‘accessors’ (to get and set values) for [RnaSeqParam](#) objects within the **easyRNASeq** package. Implemented are:

- `annotParam`
- `bamParam`
- `countBy`
- `datasource`
- `paired`
- `precision`
- `stranded`
- `strandProtocol`
- `yieldSize`

**Usage**

```
## S4 method for signature 'RnaSeqParam'  
yieldSize(object)
```

**Arguments**

object            An object derived from class RnaSeqParam.

**Value**

The value of the corresponding slot.

**Author(s)**

Nicolas Delhomme

**See Also**

- The [AnnotParam](#) class
- The [BamParam](#) class
- The [RnaSeqParam](#) class

The [BamParam](#) `yieldSize` accessor

**Examples**

```
## create the necessary AnnotParam  
annotParam <- AnnotParam(  
  datasource=system.file(  
    "extdata",  
    "Dmel-mRNA-exon-r5.52.gff3",  
    package="RnaSeqTutorial")  
)  
  
## create the RnaSeqParam  
rsp <- RnaSeqParam(annotParam=annotParam)  
## get the yieldSize Parameter  
ysize <- yieldSize(rsp)
```

---

easyRNASeq RnaSeqParam constructor

*RnaSeqParam constructor*

---

**Description**

This constructs a [RnaSeqParam](#) object, that combines all the necessary parameters for the analysis of RNA-Seq data. As much as possible, these parameters are determined automa-gi/ti-cally. It describes three sets of parameters:

- parameters describing the annotation
- parameters describing the BAM files, *i.e.* the type of sequencing that was conducted.
- parameters describing how the counting should be done.

The first two are provided through specific objects: [AnnotParam](#) and [BamParam](#) respectively. The third one is a set constituted of:

- `countBy`: the feature per which the counts should be summarized ( exon, transcript or gene. A fourth possibility - feature - can be used to define arbitrary genomic loci)
- `precision`: the precision at which the counts should be performed: bp or reads. bp used to be the default in the easyRNASeq package, whereas now reads is, following the Bioconductor main stream development.

The default parameters for the [BamParam](#) parameter are derived from the currently most common RNA-Seq experimental use-case: strand-specific paired-end Illumina sequencing. See the respective manual pages of [AnnotParam](#) and [BamParam](#) for more details.

## Usage

```
## S4 method for signature 'ANY'
RnaSeqParam(annotParam = AnnotParam(),
  bamParam = BamParam(), countBy = c("exons", "features", "genes",
  "transcripts"), precision = c("read", "bp"))
```

## Arguments

<code>annotParam</code>	An object derived from class <code>AnnotParam</code> .
<code>bamParam</code>	An object derived from class <code>BamParam</code> .
<code>countBy</code>	TODO
<code>precision</code>	A character value, either 'read' or 'bp' that defines the precision at which counting is done, either per read or per covered bp. 'read' is the default.

## Examples

```
annotParam <- AnnotParam(
  datasource=system.file(
    "extdata",
    "Dmel-mRNA-exon-r5.52.gff3",
    package="RnaSeqTutorial"))

## create the RnaSeqParam
rsp <- RnaSeqParam(annotParam=annotParam)

## change some defaults
RnaSeqParam(countBy="features", annotParam=annotParam)
RnaSeqParam(bamParam=BamParam(stranded=TRUE, yieldSize=1L), annotParam=annotParam)
```



**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**

<code>obj</code>	An object derived from class <a href="#">RNAseq</a> , can be a matrix for RPKM, see details
<code>force</code>	For <code>islandCount</code> , force RNAseq to redo <code>findIsland</code>
<code>from</code>	either "exons" or "features" can be used to summarize per transcript
<code>summarization</code>	Method use for summarize genes
<code>...</code>	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:
library(curl)
library(BSgenome.Dmelanogaster.UCSC.dm3)

# get the example data files - we retrieve a set of example bam files
# from GitHub using curl, as well as their index.
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

# get an example annotation file - we retrieve it from GitHub using curl
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
  "master/tutorial/easyRNASeq/gAnnot.rda"), "gAnnot.rda"))

# create an RNAseq object
# summarizing 2 bam files by exons
rnaSeq <- easyRNASeq(".",
  organism="Dmelanogaster",
  chr.sizes=seqlengths(Dmelanogaster),
  readLength=36L,
  annotationMethod="rda",
  annotationFile="gAnnot.rda",
  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

filesDirectory	The directory where the files to be used are located. Defaults to the current directory.
organism	A character string describing the organism
chr.sizes	A vector or a list containing the chromosomes' size of the selected organism or simply the string "auto". See details.
readLength	The read length in bp
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.
annotationFile	The location (full path) of the annotation file
annotationObject	A <a href="#">RangedData</a> or <a href="#">GRangesList</a> object containing the annotation.
format	The format of the reads, one of "aln", "bam". If not "bam", all the types supported by the <b>ShortRead</b> package are supported too. As of version 1.3.5, it defaults to bam.
gapped	Is the bam file provided containing gapped alignments?
count	The feature used to summarize the reads. One of 'exons', 'features', 'genes', 'islands' or 'transcripts'. See details.
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\$"
filenames	The name, not the path, of the files to use

<code>nbCore</code>	defines how many CPU core to use when computing the geneModels. Use the default parallel library
<code>filter</code>	The filter to be applied when loading the data using the "aln" format
<code>type</code>	The type of data when using the "aln" format. See the ShortRead library.
<code>chr.sel</code>	A vector of chromosome names to subset the final results.
<code>summarization</code>	A character defining which method to use when summarizing reads by genes. So far, only "geneModels" is available.
<code>normalize</code>	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!
<code>max.gap</code>	When computing read islands, the maximal gap size allowed between two islands to merge them
<code>min.cov</code>	When computing read islands, the minimal coverage to take into account for calling an island
<code>min.length</code>	The minimal size an island should have to be kept
<code>plot</code>	Whether or not to plot assessment graphs.
<code>conditions</code>	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.
<code>validity.check</code>	Shall UCSC chromosome name convention be enforced? This is only supported for a set of organisms, which are Dmelanogaster, Hsapiens, Mmusculus and Rnorvegicus; otherwise the argument 'chr.map' can be used to complement it.
<code>chr.map</code>	A data.frame describing the mapping of original chromosome names towards wished chromosome names. See details.
<code>ignoreWarnings</code>	set to TRUE (bad idea! they have a good reason to be there) if you do not want warning messages.
<code>silent</code>	set to TRUE if you do not want messages to be printed out.
<code>...</code>	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.

- `chr.map` The `chr.map` argument for the `easyRNASeq` function only works for an "organism-Name" 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 [RangedSummarizedExperiment](#), a [DESeq::newCountDataset](#), a [edgeR::DGEList](#) or [RNAseq](#).

### Author(s)

Nicolas Delhomme

### See Also

[RNAseq RangedSummarizedExperiment](#) [edgeR::DGEList](#) [DESeq::newCountDataset](#) [ShortRead::readAligned](#)

### Examples

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

# get the example data files - we retrieve a set of example bam files
# from GitHub using curl, as well as their index.
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

# get an example annotation file - we retrieve it from GitHub using curl
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
  "master/tutorial/easyRNASeq/gAnnot.rda"), "gAnnot.rda"))

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

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

## End(Not run)

```

---

easyRNASeq-datasets      *Dataset included in the package*

---

## Description

The package contains a dataset from the *Robinson, Delhomme et al., 2014* publication.

- RobinsonDelhomme2014a normalised expression count table. This dataset was generated from 17 *Populus tremula* - Eurasian aspen - trees used to assess the sexual dimorphism of this dioecious species. This count matrix has been generating following published pre-processing guidelines - see <http://www.epigenesys.eu/en/protocols/bio-informatics/1283-guidelines-for-rna-seq> - and the resulting HTSeq files have been collated and the obtained raw count matrix submitted to a variance stabilising transformation. Subsequently, the values have been transformed so that the minimal vst values - that corresponds to an absence of expression - is 0. Hence the counts in the matrix are library-size normalized, variance stabilised expression values, with a minimal value of 0.

---

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 <a href="#">DGEList</a>
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)
```

---

file.exists methods    *Extend the file.exists function to check the path slot of a Rsamtools BamFile class for existence*

---

**Description**

Check if the bam file represented by a [BamFile](#) object exists.

**Usage**

```
## S4 method for signature 'BamFile'
file.exists(...)
```

**Arguments**

...	a <a href="#">BamFile</a> object
-----	----------------------------------

## Methods

**list("signature(object = \"BamFile\")**) Checkk if the bam file linked to by a [BamFile](#) object exists.

---

genomeIntervals additional methods

*Extension for the genomeIntervals package*

---

## Description

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

## Usage

```
## S4 method for signature 'Genome_intervals'  
type(x)
```

## Arguments

x                    An object of class [Genome\\_intervals](#)

## Value

**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

```
library(curl)  
library(genomeIntervals)  
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",  
                              "master/tutorial/easyRNASeq/Dmel-mRNA-exon-r5.52.gff3.gz"),  
                              "Dmel-mRNA-exon-r5.52.gff3.gz"))  
annot<-readGff3("Dmel-mRNA-exon-r5.52.gff3.gz",quiet=TRUE)  
type(annot)
```



---

getBamFileList	<i>Get a BamFileList from a list of filenames</i>
----------------	---

---

### Description

A utility function to create a linkS4class{BamFileList-class}BamFileList object from a set of filenames. The filenames need to contain the file path if they are not in the working directory.

### Usage

```
## S4 method for signature 'character'
getBamFileList(filenames = character(0))
```

### Arguments

filenames          a character vector containing fully defined filenames

### Value

a linkS4class{BamFileList-class}BamFileList

### See Also

linkS4class{BamFileList-class}BamFileList [dir](#)

### Examples

```
library(curl)
# get the example data files - we retrieve a set of example bam files
# from GitHub using curl, as well as their index.
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

# creating a BamFileList using a directory and pattern
bfl <- getBamFileList(dir(".", pattern="[A,C,T,G]{6}\\\\.bam$",
  full.names=TRUE))

# using filenames
filenames <- dir(".", pattern="[A,C,T,G]{6}\\\\.bam$", full.names=TRUE)
bfl <- getBamFileList(filenames)

# get them recursively
filenames <- dir(path=".", pattern="[A,C,T,G]{6}\\\\.bam$",
  full.names=TRUE, recursive=TRUE)
bfl <- getBamFileList(filenames)
```

---

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 [IRangesList](#) object, where the split is performed by seqnames (e.g. chromosomes).

## Author(s)

Nicolas Delhomme

## Examples

```
## Not run:  
library("RnaSeqTutorial")  
  
obj <- getAnnotation(  
  AnnotParam(  
    organism="Dmelanogaster",  
    datasource=system.file(  
      "extdata",  
      "Dmel-mRNA-exon-r5.52.gff3",  
      package="RnaSeqTutorial"),  
    type="gff3"  
  )  
)  
  
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 = list(), fun = NULL,  
            nnodes = 1, ...)
```

## Arguments

obj	the object which processing has to be parallelizes
fun	the function to be applied in parallel
nnodes	the number of nodes to use
...	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	<i>Pretty print the content of classes from the easyRNASeq package.</i>
---------------	---

---

### Description

Print information about a [RNAseq](#), [AnnotParam](#), [BamParam](#) or [RnaSeqParam](#) object.

### Usage

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

### Arguments

x	An object from class <a href="#">RNAseq</a> , <a href="#">AnnotParam</a> , <a href="#">BamParam</a> or <a href="#">RnaSeqParam</a>
verbose	A logical to have a verbose or not output. Default to FALSE For object of the <a href="#">RNAseq</a> class only.
...	Additional arguments, currently unused.

### Value

Print information about the provided object.

### Author(s)

Nicolas Delhomme

---

RNAseq class	<i>Class "RNAseq"</i>
--------------	-----------------------

---

### Description

A class holding all the necessary information and annotation to summarize counts (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")
```

---

RnaSeqParam class	<i>Class "RnaSeqParam"</i>
-------------------	----------------------------

---

**Description**

A class holding all the necessary parameters to process a bam file issued from an RNA-Seq experiment together with the related annotation to compute a count-table using the [simpleRNASeq](#) function. The precision slot is used to determine the count unit:

- `readsdefault`. The standard [GenomicAlignments](#) [summarizeOverlaps](#) function is used to extract the read counts
- `bp`The [easyRNASeq](#) [summarization](#) functions are used to extract the read covered by counts

**Objects from the Class**

Objects can be created by calls of the form `new("RnaSeqParam", ...)` or using the `RnaSeqParam` constructor.

**Author(s)**

Nicolas Delhomme

**See Also**

- [RnaSeqParam](#) constructor
- [RnaSeqParam](#) accessors
- [simpleRNASeq](#) function
- [AnnotParam](#)
- [AnnotParam](#) constructor

- [BamParam](#)
- [BamParam constructor](#)
- [summarizeOverlaps](#)
- [easyRNASeq summarization functions](#)

## Examples

```
showClass("RnaSeqParam")
```

---

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, mc.cores=1L)
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

<code>obj</code>	An object derived from class <a href="#">AlignedRead</a>
<code>barcodes</code>	A character vector describing the multiplex (i.e. barcode) sequences used in the experiment.
<code>barcodes.qty</code>	An integer describing the number of barcodes
<code>barcode.length</code>	An integer describing the barcode length in bp
<code>edition.dist</code>	The maximal edition distance (i.e. the number of changes to apply), to accept an incorrectly sequenced barcode.
<code>type</code>	The type of barcode used. <code>independent</code> represents barcodes generated by the illumina protocol; i.e. a separate additional sequencing step performed once the first mate has been sequenced. <code>within</code> represents barcodes that are part of the sequenced reads as established by Lefrancois P et al., BMC Genomics, 2009
<code>index.only</code>	simply return the index and not the barcode themselves.
<code>mc.cores</code>	A parameter ultimately passed to <code>srdistance</code> to enable parallel processing on <code>mc.cores</code> . On linux and Mac only, windows task remain serially processed.
<code>.name</code>	An internal string describing the filter
<code>show.barcode</code>	An integer specifying how many barcodes should be displayed in the final output.
<code>...</code>	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. It supports multicore processing but has a default serial behaviour.
- `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","TTGCCA")

invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
  "master/tutorial/easyRNASeq/multiplex_export.txt.gz"),
  "multiplex_export.txt.gz"))

# the multiplexed data
alns <- readAligned(".",
  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 classes from the easyRNASeq package.*


---

## Description

Display the content of a [RNAseq](#), [AnnotParam](#), [BamParam](#) or [RnaSeqParam](#) object.

## Usage

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

## Arguments

object            An object of the [AnnotParam](#), [BamParam](#), [RnaSeqParam](#) or [RNAseq](#) class

## Methods

**list("signature(object = \"RNAseq\")")** Display the values of the different slots of the [RNAseq](#) object.

**Annot/Bam/RnaSeqParam** The respective object settings.



---

simpleRNASeq, BamFileList, RnaSeqParam-method  
*simpleRNASeq method*

---

## Description

This function is a wrapper around the more low level functionalities of the package. It is the simplest way to get a [RangedSummarizedExperiment](#) object from a set of bam files. [RangedSummarizedExperiment](#) are containers meant to hold any Next-Generation Sequencing experiment results and metadata. The simpleRNASeq method replaces the [easyRNASeq](#) function to simplify the usability. It does the following:

- use [GenomicAlignments](#) for reading/pre-processing the BAM files.
- get the [annotations](#) depending on the selected parameters
- calculate the coverage from the provided file(s)
- [summarizes](#) the read counts according to the selected summarization
- returns a [RangedSummarizedExperiment](#) object.

## Usage

```
## S4 method for signature 'BamFileList,RnaSeqParam'  
simpleRNASeq(bamFiles = BamFileList(),  
            param = RnaSeqParam(), nnodes = 1, verbose = TRUE, override = FALSE)
```

## Arguments

bamFiles	a <a href="#">BamFileList</a> object
param	RnaSeqParam a <a href="#">RnaSeqParam</a> object that describes the RNA-Seq experimental setup.
nnodes	The number of CPU cores to use in parallel
verbose	a logical to be report progress or not.
override	Should the provided parameters override the detected ones

## Value

returns a [RangedSummarizedExperiment](#) object.

## Author(s)

Nicolas Delhomme

## See Also

- For the input:
  - [AnnotParam](#)
  - [BamParam](#)
  - [RnaSeqParam](#)
- For the output: [RangedSummarizedExperiment](#)
- For related functions:
  - [BamFile](#)
  - [BamFileList](#) [getBamFileList](#)

**Examples**

```

# the data
library(curl)

# get the example data files - we retrieve a set of example bam files
# from GitHub using curl, as well as their index.
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

# and some annotation
invisible(curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
  "master/tutorial/easyRNASeq/Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"),
  "Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz"))

# get the BamFileList
bamFiles <- getBamFileList(dir(".", pattern="^[A,T].*\\.bam$", full.names=TRUE))

# create the AnnotParam
annotParam <- AnnotParam("Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz",
  type="gtf")

# create the RnaSeqParam
rnaSeqParam <- RnaSeqParam(annotParam=annotParam)

# get a RangedSummarizedExperiment containing the counts table
sexp <- simpleRNASeq(
  bamFiles=bamFiles,
  param=rnaSeqParam,
  verbose=TRUE
)

# get the counts
assays(sexp)$exons

```

---

validate, BamFile-method

*Extension of the Rsamtools package*

---

**Description**

Describes extensions to the Rsamtools package.

- For [BamFile](#) and [BamFileList](#) objects:
  - validate validates a [BamFile](#) or [BamFileList](#) object.

**Usage**

```

## S4 method for signature 'BamFile'
validate(obj, header = TRUE, cross.validation = TRUE)

```

**Arguments**

obj	An object of the <a href="#">BamFile</a> or <a href="#">BamFileList</a> class
header	a boolean to (de)activate the check for a BAM header
cross.validation	a boolean - only valid for <a href="#">BamFileList</a> objects - to (de)activate the cross validation of all the BAM files header

**Details**

validate checks whether the BAM file exists and if a BAI index is present.

**Value**

validate returns invisibly a vector of boolean. Fails anyway if any file is missing.

**Author(s)**

Nicolas Delhomme

**See Also**

- [BamFile](#)
- [BamFileList](#)

**Examples**

```
library(curl)
invisible(sapply(c("ACACTG", "ACTAGC"), function(bam){
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam"), paste0(bam, ".bam"))
  curl_download(paste0("https://github.com/UPSCb/UPSCb/raw/",
    "master/tutorial/easyRNASeq/", bam, ".bam.bai"), paste0(bam, ".bam.bai"))
}))

filenames <- dir(".", pattern="[A,C,T,G]{6}\\\\.bam$", full.names=TRUE)

bfl <- BamFileList(filenames, index=filenames)

validate(bfl)
```

# Index

## \*Topic **classes**

- AnnotParam class, [2](#)
- BamParam class, [3](#)
- RNAseq class, [36](#)
- RnaSeqParam class, [37](#)

## \*Topic **connection**

- easyRNASeq annotation methods, [9](#)
- easyRNASeq island methods, [19](#)

## \*Topic **data**

- easyRNASeq annotation methods, [9](#)
- easyRNASeq island methods, [19](#)
- easyRNASeq-datasets, [30](#)

## \*Topic **manip**

- easyRNASeq accessors, [8](#)
- easyRNASeq AnnotParam accessors, [10](#)
- easyRNASeq BamParam accessors, [12](#)
- easyRNASeq RnaSeqParam accessors, [22](#)

## \*Topic **methods**

- basename methods, [4](#)
- createSyntheticTranscripts, AnnotParamCharacter-method, [5](#)
- DESeq additional methods, [7](#)
- easyRNASeq annotation methods, [9](#)
- easyRNASeq correction methods, [14](#)
- easyRNASeq coverage methods, [16](#)
- easyRNASeq GenomicRanges package extension, [18](#)
- easyRNASeq island methods, [19](#)
- easyRNASeq summarization methods, [24](#)
- easyRNASeq, character-method, [26](#)
- edgeR additional methods, [30](#)
- file.exists methods, [31](#)
- IRanges additional methods, [34](#)
- parallel additional methods, [35](#)
- print methods, [36](#)
- ShortRead additional methods, [38](#)
- show methods, [40](#)
- simpleRNASeq, BamFileList, RnaSeqParam-method, [41](#)
- validate, BamFile-method, [42](#)
- easyRNASeq package, [21](#)
- .geneModelSummarization, [25, 26](#)
- accessors (easyRNASeq accessors), [8](#)
- alignData (ShortRead additional methods), [38](#)
- AlignedRead, [38, 39](#)
- annotations, [41](#)
- AnnotParam, [3, 4, 6, 9–11, 23, 24, 36, 37, 40, 41](#)
- AnnotParam (easyRNASeq AnnotParam constructor), [11](#)
- annotParam (easyRNASeq RnaSeqParam accessors), [22](#)
- AnnotParam class, [2](#)
- AnnotParam, character-method (easyRNASeq AnnotParam constructor), [11](#)
- AnnotParam, GRanges-method (easyRNASeq AnnotParam constructor), [11](#)
- AnnotParam, missing-method (easyRNASeq AnnotParam constructor), [11](#)
- AnnotParam, RangedData-method (easyRNASeq AnnotParam constructor), [11](#)
- annotParam, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)
- AnnotParam-accessors (easyRNASeq AnnotParam accessors), [10](#)
- AnnotParam-class (AnnotParam class), [2](#)
- AnnotParamCharacter, [5](#)
- AnnotParamCharacter-class (AnnotParam class), [2](#)
- AnnotParamObject-class (AnnotParam class), [2](#)
- assay (easyRNASeq package), [21](#)
- BamFile, [4, 31, 32, 41–43](#)
- BamFileList, [4, 21, 41–43](#)
- BamFileList (easyRNASeq package), [21](#)
- BamFileList-class (easyRNASeq package), [21](#)
- BamParam, [3, 12, 13, 23, 24, 36, 38, 40, 41](#)

## \*Topic **package**

- BamParam (easyRNASeq BamParam constructor), 13
- bamParam (easyRNASeq RnaSeqParam accessors), 22
- BamParam class, 3
- BamParam, ANY-method (easyRNASeq BamParam constructor), 13
- bamParam, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 22
- BamParam-accessors (easyRNASeq BamParam accessors), 12
- BamParam-class (BamParam class), 3
- barcodePlot (ShortRead additional methods), 38
- barcodePlot, AlignedRead-method (ShortRead additional methods), 38
- barcodePlot, DNASTringSet-method (ShortRead additional methods), 38
- barcodePlot, ShortReadQ-method (ShortRead additional methods), 38
- basename (basename methods), 4
- basename methods, 4
- basename, BamFile-method (basename methods), 4
- basename, BamFileList-method (basename methods), 4
- BiocParallel, 21
- biomaRt, 21
- Biostrings, 11, 21
- BSgenome, 11, 21
  
- chastityFilter (ShortRead additional methods), 38
- chastityFilter, SRFilter-method (ShortRead additional methods), 38
- chromosomeFilter (easyRNASeq package), 21
- chrSize (easyRNASeq accessors), 8
- chrSize, RNAseq-method (easyRNASeq accessors), 8
- chrSize<- (easyRNASeq accessors), 8
- chrSize<-, RNAseq, integer-method (easyRNASeq accessors), 8
- chrSize<-, RNAseq, list-method (easyRNASeq accessors), 8
- clusterApply, 35
- colnames, 18
- colnames (easyRNASeq GenomicRanges package extension), 18
- colnames, GRanges-method (easyRNASeq GenomicRanges package extension), 18
- colnames, GRangesList-method (easyRNASeq GenomicRanges package extension), 18
- compose (easyRNASeq package), 21
- countBy (easyRNASeq RnaSeqParam accessors), 22
- countBy, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 22
- CountDataSet, 7, 8, 21
- createSyntheticTranscripts (createSyntheticTranscripts, AnnotParamCharacter), 5
- createSyntheticTranscripts, AnnotParamCharacter-method, 5
- createSyntheticTranscripts, character-method (createSyntheticTranscripts, AnnotParamCharacter), 5
  
- DataFrame, 18
- datasource (easyRNASeq AnnotParam accessors), 10
- datasource, AnnotParam-method (easyRNASeq AnnotParam accessors), 10
- datasource, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 22
- Defunct functions, 6
- demultiplex (ShortRead additional methods), 38
- demultiplex, AlignedRead-method (ShortRead additional methods), 38
- demultiplex, DNASTringSet-method (ShortRead additional methods), 38
- demultiplex, ShortReadQ-method (ShortRead additional methods), 38
- DESeq, 21
- DESeq additional methods, 7
- DESeq estimateDispersions, 28
- DESeq methods, 21
- DESeq: newCountDataset, 29
- DGEList, 31
- dir, 33
  
- easyRNASeq, 6, 7, 21, 41

- easyRNASeq (Defunct functions), 6
- easyRNASeq accessors, 8
- easyRNASeq annotation methods, 9, 21
- easyRNASeq AnnotParam accessors, 10
- easyRNASeq AnnotParam constructor, 11
- easyRNASeq BamParam accessors, 12
- easyRNASeq BamParam constructor, 13
- easyRNASeq correction methods, 14, 21
- easyRNASeq coverage methods, 16, 21
- easyRNASeq defunct annotation methods, 17
- easyRNASeq GenomicRanges package extension, 18
- easyRNASeq island methods, 19
- easyRNASeq package, 21
- easyRNASeq package-package (easyRNASeq package), 21
- easyRNASeq RnaSeqParam accessors, 22
- easyRNASeq RnaSeqParam constructor, 23
- easyRNASeq summarization methods, 21, 24
- easyRNASeq, character-method, 26
- easyRNASeq, RNAseq-method (Defunct functions), 6
- easyRNASeq-datasets, 30
- easyRNASeq-defunct (easyRNASeq, character-method), 26
- easyRNASeq-package (easyRNASeq package), 21
- edgeR, 21
- edgeR additional methods, 30
- edgeR methods, 21
- edgeR:DGEList, 29
- exonCounts (easyRNASeq summarization methods), 24
- exonCounts, RNAseq-method (easyRNASeq summarization methods), 24
- featureCounts (easyRNASeq summarization methods), 24
- featureCounts, RNAseq-method (easyRNASeq summarization methods), 24
- fetchAnnotation (Defunct functions), 6
- fetchAnnotation-defunct (easyRNASeq defunct annotation methods), 17
- fetchCoverage, 6, 7
- fetchCoverage (Defunct functions), 6
- fetchCoverage, RNAseq-method (Defunct functions), 6
- fetchCoverage-deprecated (easyRNASeq coverage methods), 16
- file.exists (file.exists methods), 31
- file.exists methods, 31
- file.exists, BamFile-method (file.exists methods), 31
- fileName (easyRNASeq accessors), 8
- fileName, RNAseq-method (easyRNASeq accessors), 8
- fileName<- (easyRNASeq accessors), 8
- fileName<-, RNAseq-method (easyRNASeq accessors), 8
- findIslands, 25, 26
- findIslands (easyRNASeq island methods), 19
- findIslands, RNAseq-method (easyRNASeq island methods), 19
- fitInfo, 7
- GAlignments, 18
- geneCounts (easyRNASeq summarization methods), 24
- geneCounts, RNAseq-method (easyRNASeq summarization methods), 24
- geneModel (easyRNASeq accessors), 8
- geneModel, RNAseq-method (easyRNASeq accessors), 8
- geneModel<- (easyRNASeq accessors), 8
- geneModel<-, RNAseq-method (easyRNASeq accessors), 8
- Genome\_intervals, 6, 32
- genomeIntervals, 21
- genomeIntervals additional methods, 32
- genomeIntervals object, 32
- GenomicAlignments, 41
- genomicAnnotation (easyRNASeq accessors), 8
- genomicAnnotation, RNAseq-method (easyRNASeq accessors), 8
- genomicAnnotation<- (easyRNASeq accessors), 8
- genomicAnnotation<-, RNAseq-method (easyRNASeq accessors), 8
- GenomicFeatures, 21
- GenomicRanges, 18, 21
- getAnnotation, 11, 12, 17
- getAnnotation (easyRNASeq annotation methods), 9
- getAnnotation, AnnotParam-method (easyRNASeq annotation methods), 9
- getBamFileList, 33, 41
- getBamFileList, character-method (getBamFileList), 33
- getBM, 9, 28
- GRanges, 6, 9, 11, 12, 18

- GRangesList, [18, 27, 34](#)
- hist, [20](#)
- IRanges, [21](#)
- IRanges (easyRNASeq package), [21](#)
- IRanges additional methods, [34](#)
- IRangesList, [34](#)
- islandCounts (easyRNASeq summarization methods), [24](#)
- islandCounts, RNAseq-method (easyRNASeq summarization methods), [24](#)
- knownOrganisms (Defunct functions), [6](#)
- knownOrganisms-defunct (easyRNASeq defunct annotation methods), [17](#)
- librarySize (easyRNASeq accessors), [8](#)
- librarySize, RNAseq-method (easyRNASeq accessors), [8](#)
- librarySize<- (easyRNASeq accessors), [8](#)
- librarySize<-, RNAseq-method (easyRNASeq accessors), [8](#)
- list.files, [28](#)
- listDatasets, [9](#)
- locfit (DESeq additional methods), [7](#)
- lp (DESeq additional methods), [7](#)
- makePSOCKcluster, [35](#)
- multivariateConditions (DESeq additional methods), [7](#)
- multivariateConditions, CountDataSet-method (DESeq additional methods), [7](#)
- naPositionFilter (ShortRead additional methods), [38](#)
- naPositionFilter, SRFilter-method (ShortRead additional methods), [38](#)
- newCountDataSet (DESeq additional methods), [7](#)
- nFilter (easyRNASeq package), [21](#)
- optionally apply, [27](#)
- organismName (Defunct functions), [6](#)
- organismName, RNAseq-method (Defunct functions), [6](#)
- organismName<- (Defunct functions), [6](#)
- organismName<-, RNAseq-method (Defunct functions), [6](#)
- paired (easyRNASeq BamParam accessors), [12](#)
- paired, BamParam-method (easyRNASeq BamParam accessors), [12](#)
- paired, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)
- parallel, [21](#)
- parallel additional methods, [35](#)
- parallelize (parallel additional methods), [35](#)
- parallelize, BamFileList, function-method (parallel additional methods), [35](#)
- parallelize, GRangesList, function-method (parallel additional methods), [35](#)
- parallelize, list, function-method (parallel additional methods), [35](#)
- parallelize, vector, function-method (parallel additional methods), [35](#)
- plot.default, [7](#)
- plotBCV, [7](#)
- plotDispersionEstimates (DESeq additional methods), [7](#)
- plotDispersionEstimates, CountDataSet-method (DESeq additional methods), [7](#)
- plotDispersionEstimates, DGEList-method (Defunct functions), [6](#)
- plotDispEsts, [7, 8](#)
- plotDispLSD (DESeq additional methods), [7](#)
- plotDispLSD, CountDataSet-method (DESeq additional methods), [7](#)
- plotNormalizationFactors (edgeR additional methods), [30](#)
- plotNormalizationFactors, DGEList, character, character-method (edgeR additional methods), [30](#)
- precision (easyRNASeq RnaSeqParam accessors), [22](#)
- precision, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)
- print, [37](#)
- print (print methods), [36](#)
- print methods, [36](#)
- print, AnnotParam-method (print methods), [36](#)
- print, BamParam-method (print methods), [36](#)
- print, RNAseq-method (print methods), [36](#)
- print, RnaSeqParam-method (print methods), [36](#)

- RangedData, [11](#), [12](#), [21](#), [27](#), [34](#), [37](#)
- RangedData (easyRNASeq package), [21](#)
- RangedData-class (easyRNASeq package), [21](#)
- RangedSummarizedExperiment, [21](#), [29](#), [41](#)
- RangedSummarizedExperiment-class (easyRNASeq package), [21](#)
- ranges (IRanges additional methods), [34](#)
- ranges, RNAseq-method (IRanges additional methods), [34](#)
- readCounts, [14](#), [15](#)
- readCounts (easyRNASeq accessors), [8](#)
- readCounts, RNAseq-method (easyRNASeq accessors), [8](#)
- readCounts<- (easyRNASeq accessors), [8](#)
- readCounts<- , RNAseq-method (easyRNASeq accessors), [8](#)
- readCoverage (easyRNASeq accessors), [8](#)
- readCoverage, RNAseq-method (easyRNASeq accessors), [8](#)
- readCoverage<- (easyRNASeq accessors), [8](#)
- readCoverage<- , RNAseq-method (easyRNASeq accessors), [8](#)
- readGff3, [9](#)
- readGffGtf, [9](#), [28](#)
- readIslands (easyRNASeq accessors), [8](#)
- readIslands, RNAseq-method (easyRNASeq accessors), [8](#)
- readIslands<- (easyRNASeq accessors), [8](#)
- readIslands<- , RNAseq-method (easyRNASeq accessors), [8](#)
- readLength (easyRNASeq accessors), [8](#)
- readLength, RNAseq-method (easyRNASeq accessors), [8](#)
- readLength<- (easyRNASeq accessors), [8](#)
- readLength<- , RNAseq-method (easyRNASeq accessors), [8](#)
- Rle, [17](#)
- RleList, [37](#)
- RNAseq, [14](#), [16](#), [21](#), [25](#), [29](#), [34](#), [36](#), [40](#)
- RNAseq (RNAseq class), [36](#)
- RNAseq class, [36](#)
- RNAseq-class (RNAseq class), [36](#)
- RnaSeqParam, [3](#), [4](#), [22](#), [23](#), [36](#), [40](#), [41](#)
- RnaSeqParam (easyRNASeq RnaSeqParam constructor), [23](#)
- RnaSeqParam class, [37](#)
- RnaSeqParam, ANY-method (easyRNASeq RnaSeqParam constructor), [23](#)
- RnaSeqParam-accessors (easyRNASeq RnaSeqParam accessors), [22](#)
- RnaSeqParam-class (RnaSeqParam class), [37](#)
- RobinsonDelhomme2014 (easyRNASeq-datasets), [30](#)
- RPKM (easyRNASeq correction methods), [14](#)
- RPKM, matrix, ANY, vector, vector-method (easyRNASeq correction methods), [14](#)
- RPKM, RNAseq, ANY, ANY, ANY-method (easyRNASeq correction methods), [14](#)
- RPKM, RNAseq-method (easyRNASeq correction methods), [14](#)
- Rsamtools, [21](#)
- seqnames, RNAseq-method (easyRNASeq accessors), [8](#)
- ShortRead, [21](#)
- ShortRead additional methods, [38](#)
- ShortRead methods, [21](#)
- ShortRead:readAligned, [17](#), [29](#)
- show methods, [40](#)
- show, AnnotParam-method (show methods), [40](#)
- show, BamParam-method (show methods), [40](#)
- show, RNAseq-method (show methods), [40](#)
- show, RnaSeqParam-method (show methods), [40](#)
- simpleRNASeq, [7](#)
- simpleRNASeq (simpleRNASeq, BamFileList, RnaSeqParam-method), [41](#)
- simpleRNASeq, BamFileList, RnaSeqParam-method, [41](#)
- SRFilter, [39](#)
- SRFilterResult (easyRNASeq package), [21](#)
- stopCluster, [35](#)
- stranded (easyRNASeq BamParam accessors), [12](#)
- stranded, BamParam-method (easyRNASeq BamParam accessors), [12](#)
- stranded, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)
- strandProtocol (easyRNASeq BamParam accessors), [12](#)
- strandProtocol, BamParam-method (easyRNASeq BamParam accessors), [12](#)
- strandProtocol, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)
- summarizeOverlaps, [38](#)
- summarizes, [41](#)



transcriptCounts (easyRNASeq summarization methods), [24](#)  
transcriptCounts, RNAseq-method (easyRNASeq summarization methods), [24](#)  
type (easyRNASeq package), [21](#)  
type, AnnotParam-method (easyRNASeq AnnotParam accessors), [10](#)  
type, Genome\_intervals-method (genomeIntervals additional methods), [32](#)

unsafeAppend (easyRNASeq GenomicRanges package extension), [18](#)  
unsafeAppend, GAlignments, GAlignments-method (easyRNASeq GenomicRanges package extension), [18](#)

validate (validate, BamFile-method), [42](#)  
validate, BamFile-method, [42](#)  
validate, BamFileList-method (validate, BamFile-method), [42](#)

yieldSize (easyRNASeq BamParam accessors), [12](#)  
yieldSize, BamParam-method (easyRNASeq BamParam accessors), [12](#)  
yieldSize, RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), [22](#)