Package ‘easyRNASeq’

January 15, 2026

Version 2.47.0

Date 2025-06-23

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 (>= 2.64.0), BiocFileCache (>= 2.12.0), BiocGenerics (>= 0.50.0), BiocParallel (>= 1.38.0), biomaRt (>= 2.60.1), Biostrings (>= 2.77.2), edgeR (>= 4.2.1), Seqinfo, genomeIntervals (>= 1.60.0), GenomicAlignments (>= 1.45.1), GenomicRanges (>= 1.61.1), SummarizedExperiment (>= 1.39.1), graphics, IRanges (>= 2.38.1), LSD (>= 4.1-0), methods, parallel, rappdirs (>= 0.3.3), Rsamtools (>= 2.25.1), S4Vectors (>= 0.42.1), ShortRead (>= 1.62.0), utils

Suggests BiocStyle (>= 2.32.1), BSgenome (>= 1.72.0), BSgenome.Dmelanogaster.UCSC.dm3 (>= 1.4.0), curl, knitr, rmarkdown, RUnit (>= 0.4.33)

License Artistic-2.0

LazyLoad yes

VignetteBuilder knitr

biocViews GeneExpression, RNASeq, Genetics, Preprocessing, ImmunoOncology

RoxygenNote 7.3.2

Encoding UTF-8

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

git_branch devel

git_last_commit 1cb9a73

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-01-15

Contents

AnnotParam class 3
AnnotParam internal methods 3
BamParam class 4
basename methods 5
BiocFileCache methods 6
createSyntheticTranscripts,AnnotParamCharacter-method 7
Defunct functions 9
easyRNASeq accessors 9
easyRNASeq annotation internal methods 10
easyRNASeq annotation methods 11
easyRNASeq AnnotParam accessors 12
easyRNASeq AnnotParam constructor 13
easyRNASeq BamParam accessors 14
easyRNASeq BamParam constructor 15
easyRNASeq correction methods 16
easyRNASeq coverage methods 18
easyRNASeq defunct annotation methods 20
easyRNASeq GenomicRanges package extension 21
easyRNASeq internal methods 22
easyRNASeq island methods 24
easyRNASeq package 25
easyRNASeq RnaSeqParam accessors 27
easyRNASeq RnaSeqParam constructor 28
easyRNASeq summarization internal methods 29
easyRNASeq summarization methods 31
easyRNASeq,character-method 33
easyRNASeq-datasets 37
easyRNASeq-global-variables 37
edgeR additional methods 38
file.exists methods 39
genomeIntervals additional methods 40
getBamFileList 41
IRanges additional methods 42
parallel additional methods 43
print methods 44
RNAseq class 44
RnaSeqParam class 45
ShortRead additional methods 46
show methods 49
simpleRNASeq,BamFileList,RnaSeqParam-method 49
validate,BamFile-method 51

Index **53**

`AnnotParam class` *Class "AnnotParam"*

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, prefered). 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")
```

`AnnotParam internal methods`
Internal methods of AnnotParam objects

Description

These are `AnnotParam`AnnotParam class internal methods:

- .validate validate the content of an AnnotParam object

Usage

```
.validate(obj, verbose = TRUE, lenient = FALSE)
```

Arguments

`obj`	An AnnotParam object
`verbose`	To print (or not) messages
`lenient`	Relax validation parameters for gtf files
`...`	additional arguments passed to the retrieval function. At the moment only forwarded to the **biomaRt** `getBM` function.

Value

`.validate`	invisibly return a TRUE logical on success and stops on failure

Author(s)

Nicolas Delhomme

`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

Extend the basename function to display Rsamtools BamFile class basename

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.

`BiocFileCache methods` *Manages the data necessary for the examples using BiocFileCache*

Description

Manages the tutorial, example and vignette data using the `BiocFileCache` package

Usage

```
fetchData(fileURL)
tutorialData(...)
```

Arguments

`...`	unused for the time being
`fileURL`	The URL of the file to retrieve. Alternatively, the ID of the file in the BiocFileCache (i.e. the file basename), can be used.

Methods

.get_cache internal function to set up the cache

fetchData A function to fetch tutorial data, a file at a time

tutorialData the function to retrieve all the tutorial data and cache it, if it is not already available

vignetteData the function to retrieve all the tutorial data and cache it, if it is not already available

See Also

`BiocFileCache`

Examples

```
tdir <- tutorialData()
gAnnot.path <- fetchData("gAnnot.rda")
vdir <- vignetteData()
md5.txt <- fetchData("md5.txt")
```

`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 `AnnotParamCharacter` 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'

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` 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
Dm.gtf <- fetchData("Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz")

# create the AnnotParam
annotParam <- AnnotParam(
  datasource=Dm.gtf,
  type="gtf")

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

`Defunct functions` *The following function are defunct:*

- `easyRNASeq`
- `fetchCoverage`
- `fetchAnnotation`
- `knownOrganisms`

Description

- 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 occuring 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 'isUnmappedQuery' parameters were obsolete.

`easyRNASeq accessors` *Accessors for RNAseq class*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Nicolas Delhomme

Examples

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

`easyRNASeq annotation internal methods` *Internal easyRNASeq annotation methods*

Description

These are internal methods used to retrieve annotations tabularll `.getBmRange`Use `biomaRt` to get exon annotations. `.getGffRange`Use `Genome_intervals_stranded` to get annotation from a gff file. `.getGtfRange`Use `Genome_intervals_stranded` to get annotation from a gtf file. `.geneModelAnnotation`Use the provided exon annotation to define gene models. `.readGffGtf`Use `Genome_intervals_stranded` to get annotation from a gff or gtf file. It is called from `getGffRange` and `getGtfRange`.

Usage

```
.getBmRange(obj, ...)
```

Arguments

`obj`	an `AnnotParam` object containing the necessary retrieval information (datasource and type)
`...`	Additional arguments, passed to more internal functions.
`annotation.type`	describes the kind of annotation to keep the information from in a gtf or gff file. If set to NULL all the annotations are returned.
`fields`	added a parameter that allows defining the fields parsed from a gtf file. Still internal, but could easily be externalized.
`filename`	filename that contains the annotations
`format`	describes the kind of annotation provided. One of gtf or gff.
`gAnnot`	a `GRanges` object containing exon annotations
`nbCore`	number of CPU cores to use

Details

To use multicore machines more efficiently, the default parallel package will be used to parallelize the processing.

Value

A `GRanges` containing the loaded or processed annotations.

Author(s)

Nicolas Delhomme

`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** `genomeIntervals-readGff3` is used to import the data.

Usage

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

Arguments

`obj`	An object of class `AnnotParam`
`verbose`	a boolean to turn on verbosity
`...`	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

`object`	An object derived from class `AnnotParam`.
`x`	An object derived from class `AnnotParam`.

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

```
# fetch the example data
Dm.annot <- fetchData("Dmel-mRNA-exon-r5.52.gff3.gz")

annot <- AnnotParam(datasource=Dm.annot)
# 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` 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 whishes to retrieve.

Usage

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

Arguments

`datasource`	a character 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 `GRanges` 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`
- `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
# fetch the example data
gAnnot.rda <- fetchData("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 `Rsamtools package yieldSize 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 `RNAseq` 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 `RNAseq` 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 **ShortRead** 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

`x`	An object of the `GRanges` or `GRangesList` class
`do.NULL`	see `row_colnames` for details
`prefix`	see `row_colnames` for details
`obj1`	A `GAlignments` object
`obj2`	A `GAlignments` 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` `row_colnames`

Examples

```
# an example of annotation
grngs <- GRanges(seqnames=c("chr01","chr01","chr02"),
                    ranges=IRanges(
                            start=c(10,30,100),
                            end=c(21,53,123)),
                         strand=c("+","+","-"),
                         transcripts=c("trA1","trA2","trB"),
                         gene=c("gA","gA","gB"),
                         exon=c("e1","e2","e3")
                         )

# 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 internal methods`

Internal methods of RNAseq objects

Description

These are generic internal methods:

- Actual
    - .catn Just some pretty printing.
    - .checkArguments check that the provided argument match one of the formal definition of the function. Stop if not.
    - .extractIRangesList extract an IRanges object from an AlignedRead or a GAlignments object or a list returned by reading a bam file with Rsamtools. It returns a list containing the IRangesList and library size.
    - .getArguments For a given function returns the arguments passed as part of the ... that match that function formals.
    - .getName Get the genomicAnnotation object names. Necessary to deal with the different possible annotation object: `GRanges` or `GRangesList`.
    - .getWidth Get the genomicAnnotation withs. Necessary to deal with the different possible annotation object: `GRanges` or `GRangesList`.

  - .normalizationDispatcher a function to dispatch the normalization depending on the 'outputFormat' chosen by the user.
  - reduce Allow proper dispatch between the intervals and the GenomicRanges reduce function
  - strand Allow proper dispatch between the genomeIntervals and the GenomicRanges strand function
  - strand<- Allow proper dispatch between the genomeIntervals and the GenomicRanges strand replace function
- Defunct
  - .convertToUCSC convert chromosome names to UCSC compliant ones.
  - .list.files check the arguments passed through the ... to select only the valid ones (defunct).

Usage

```
.extractIRangesList(obj, chr.sel = c())
```

Arguments

`obj`	An RNAseq object, or for the 'normalizationDispatcher', depending on the type: a CountDataSet, a DGEList, a matrix, or an RNAseq object respectively
`chr.sel`	A list of chromosome to restrict the IRanges spaces returned.
`arg`	The argument name to check for.
`chr.names`	The chromosome names, as a character vector, to be converted to UCSC ones
`fun`	The name of the function
`organism`	The organism name
`type`	character string specifying the type of object (normalizationDispatcher)
`value`	the appropriate strand object (strand and strand<-) or the provided argument value (checkArguments)
`x`	an object of the GenomicRanges, intervals or genomeIntervals package
`...`	For `.getArguments` a list of named parameters to be matched against a function formal definition. For `.catn`, the values to be printed.

Value

`argString`	a character string representing these arguments and their value that matched those defined in the formal definition of the function
`convertedChrNames`	a converted vector of chromosome names
`i.range`	an IRange object
`names`	The annotation names, i.e. a combination of exon, feature, transcript and gene
`normalized.counts`	Depending on the type, a CountDataSet, a DGEList, a NumericList, or NULL respectively

Author(s)

Nicolas Delhomme

`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 GRanges 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))
  )

 # fetch the example data
 bamFilePath <- fetchData("ACACTG.bam")

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

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

## End(Not run)
```

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

Description

Offers functionalities to summarize read counts per feature of interest, e.g. exons, transcripts, genes, etc. Offers functionalities to normalize the summarized counts using a 3rd party package: `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. Using RPKM is only advisable for visualization purposes and should never be used for Differential Expression with edgeR or DESeq2. 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 `edgeR` and the corresponding 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 co`

Author(s)

Nicolas Delhomme, Bastian Schiffthaler, Ismael Padioleau

See Also

The class RNAseq specification: `RNAseq`

The default output class specification: `RangedSummarizedExperiment`

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

The suggested packages: `parallel` `GenomicFeatures`

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

- Classes `BamFileList-class` `RangedSummarizedExperiment`
- Functions/Methods `The RangedSummarizedExperiment assay accessor` The BamFileList constructor `BamFileList-class` The IRanges constructor `IRanges-constructor` For the SRFilterResult, chromosomeFilter, compose and nFilter methods`srFilter`

Examples

```
# the data
tdir <- tutorialData()

 # get the example annotation file - we retrieve a gtf file from GitHub
 annot <- fetchData("Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz")

 # create the AnnotParam
 annotParam <- AnnotParam(
   datasource=annot,
   type="gtf")

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

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

 # get the bamfiles (from the Bioc cache in this example)
filenames <- dir(tdir,pattern="[A,T].*\\.bam$",full.names=TRUE)
indexnames <- sapply(paste0(sub(".*_","",basename(filenames)),".bai"),fetchData)
bamFiles <- getBamFileList(filenames,indexnames)

# 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 sepcific 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 forth 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

`annotParam`	An object derived from class `AnnotParam`.
`bamParam`	An object derived from class `BamParam`.
`countBy`	TODO
`precision`	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)
```

`easyRNASeq summarization internal methods`

Internal count and summarization methods

Description

These are internal methods related to counting and summarizing reads

- For counting reads:
  - .doCount A dispatcher higher level function to count and summarize reads. Externalized so that it can be parallelized.
  - .doBasicCount A function to calculate the counts for 'exons' or 'features'

- For summarizing per genes: these methods are called by the method `geneCounts`. Having performed the `exonCounts` is a pre-requisite.
  - .bestExonSummarization Identify the exon showing the highest coverage.
  - .geneModelSummarization Sum the coverage values of the synthetic exons constituting a gene model.
- For managing the summarized read count structure:
  - .extendCountList extend or create the result count list of matrices

Usage

```
.bestExonSummarization(obj)
```

Arguments

`obj`	An object derived from class `RNAseq`
`chr.map`	A data.frame describing the mapping of original chromosome names towards wished chromosome names. See the details in `easyRNASeq`.
`chr.sel`	A vector of chromosome names to subset the final results.
`cList`	list of lists that contain count results
`count`	The feature used to summarize the reads. One of 'exons','features','genes','islands' or 'transcripts'.
`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. As of version 1.3.5, it defaults to bam.
`gapped`	Is the bam file provided containing gapped alignments?
`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
`plot`	Whether or not to plot assessment graphs.
`rnaSeq`	An object derived from class `RNAseq`
`summarization`	A character defining which method to use when summarizing reads by genes. So far, only "geneModels" is available.
`silent`	set to TRUE if you do not want messages to be printed out.
`subType`	character string defining a sub type of counts, i.e. for the gene type one of bestExon or geneModel
`type`	• .extendCountList: character string specifying the type of count ("exons", "transcripts", "genes" or islands) • .doCount: the type of data when using the "aln" format. See the ShortRead library.

`validity.check`	Shall UCSC chromosome name convention be enforced? This is only supported for a set of organisms, which are Dmelanogaster, Hsapiens, Mmusculus and Rnorvegicus; otherwise the argument 'chr.map' can be used to complement it.
`values`	a named vector containing count results
`...`	additional arguments. See the details in `easyRNASeq`.

Value

- .doCount: a list containing
  - counts: the summarized counts as a matrix of dimension number of genes x 1
  - size: the library size
- .doBasicCount: a vector containing read counts.
- .bestExonSummarization: a vector containing summarized counts.
- .geneModelSummarization: a vector containing summarized counts.
- .extendCountList: a named list of matrices. The names are according to the counting/summarization already performed.

Author(s)

Nicolas Delhomme

See Also

- `ShortRead:readAligned`
- `RNAseq` `easyRNASeq`.

`easyRNASeq summarization methods`

Count methods for RNAseq object

Description

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

- `exonCounts`: for that summarization, reads are summarized per exons. An "exon" field is necessary in the annotation object for this to work. See `easyRNASeq annotation methods` for more details on the annotation object.
- `featureCounts` is similar to the 'exons' one. This is just a wrapper to summarize count for genomic features that are not exon related. I.e. one could use it to measure eRNAs. Again, a "feature" field is necessary in the annotation object for this to work.
- `geneCounts` sums the counts per either `bestExons` or `geneModels`. In either case, the annotation object needs to contain both an "exon" and a "gene" field.
- `islandCounts` sums the counts per computed islands.

- `transcriptCounts` sums the counts obtained by exons into their respective transcripts. Note that this often result in counting some reads several times. For this function to work you need both an "exon" and a "transcript" field in your annotation object. To avoid this, one could create transcript specific synthetic exons, i.e. features that would be unique to a transcript. To offer this possibility, transcripts count can be summarized from "features", in which case the annotation object need to have both the "feature" and "transcript" fields defined.

Usage

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

Arguments

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

Details

. . . for

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

Value

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

Author(s)

Nicolas Delhomme

See Also

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

Examples

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

 # get the example data files
 tdir <- tutorialData()
```

```
    # get an example annotation file - we retrieve it from GitHub using curl
    gAnnot.rda <- fetchData("gAnnot.rda")

    # create an RNAseq object
    # summarizing 2 bam files by exons
    rnaSeq <- easyRNASeq(tdir,
                         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, this being strongly 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 = GRangesList(),
    format = c("bam", "aln"),
    gapped = FALSE,
    count = c("exons", "features", "genes", "islands", "transcripts"),
    outputFormat = c("matrix", "SummarizedExperiment", "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 `GRangesList` object containing the annotation.
`format`	The format of the reads, one of "aln","bam". If not "bam", all the types supported by the **ShortRead** package are supported too. As of version 1.3.5, it defaults to bam.
`gapped`	Is the bam file provided containing gapped alignments?
`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 `edgeR`, `RNAseq` or `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
`nbCore`	defines how many CPU core to use when computing the geneModels. Use the default parallel library
`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 ShortRead library.
`chr.sel`	A vector of chromosome names to subset the final results.
`summarization`	A character defining which method to use when summarizing reads by genes. So far, only "geneModels" is available.
`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 `edgeR`. Note that you should not normalize the data prior to using edgeR!
`max.gap`	When computing read islands, the maximal gap size allowed between two islands to merge them
`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
`plot`	Whether or not to plot assessment graphs.
`conditions`	A vector of descriptor, each sample must have a descriptor if you use outputFormat 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.
`validity.check`	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.
`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.
`silent`	set to TRUE if you do not want messages to be printed out.
`...`	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 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 `GRangesList` object need to be provided. Check the vignette or the examples at the bottom of this page for examples. The data.frame-like structure of these objects is where `easyRNASeq` will look for the exon, feature, transcript, or gene identifier. Depending on the count method selected, it is essential that the akin column name is present in the annotationObject. E.g. when counting "features", the annotationObject has to contain a "feature" field.
- the chr.map The chr.map argument for the easyRNASeq function only works for an "organismName" of value 'custom' with the "validity.check" parameter set to 'TRUE'. This data.frame should contain two columns named 'from' and 'to'. The row should represent the chromosome name in your original data and the wished name in the output of the function.
- count The count can be summarized by exons, features, genes, islands or transcripts. While exons, genes and transcripts are obvious, "features" describes any features provided by the user, e.g. enhancer loci. These are processed as the exons are. For "islands", it is for an under development function that identifies de-novo expression loci and count the number of reads overlapping them.
- chr.sizes If set to "auto", then the format has to be "bam", in which case the chromosome names and size are extracted from the BAM header

Value

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

Author(s)

Nicolas Delhomme

See Also

`RNAseq` `RangedSummarizedExperiment` `edgeR:DGEList` `ShortRead:readAligned`

Examples

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

# get the example data
tdir <- tutorialData()


# get an example annotation file
gAnnot.rda <- fetchData("gAnnot.rda")

# creating a count table from 4 bam files
count.table <- easyRNASeq(filesDirectory="tdir",
    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 GRangesList annotation
 grngs <- GRanges(seqnames=c("chr01","chr01","chr02"),
                     ranges=IRanges(
                             start=c(10,30,100),
                             end=c(21,53,123)),
                          strand=c("+","+","-"),
                          transcript=c("trA1","trA2","trB"),
                          gene=c("gA","gA","gB"),
                          exon=c("e1","e2","e3")
                          )

 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.

`easyRNASeq-global-variables`
Objects created when the package is attached.

Description

The package creates the following objects when attached

- GTF.FIELDS
- ANNOTATION.TYPE
- TUTORIAL.DATA
- VIGNETTE.DATA

Arguments

`libname`	a character string giving the library directory where the package defining the namespace was found.
`pkgname`	a character string giving the name of the package.

Details

These objects hold the following information

- GTF.FIELDS `c("gene_id","transcript_id","exon_id","gene_name")`
- ANNOTATION.TYPE `c(mRNA="mRNA",exon="exon")`
- TUTORIAL.DATA: The list of files needed for the help and test pages
- VIGNETTE.DATA: The list of files needed for the vignette

and are designed as global variables to expose the fact that they are hardcoded. These exist as placeholder in case a user would require different values for these.

See Also

`.onAttach` in the `base` package.

`edgeR additional methods`

Extension for the edgeR package

Description

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

Usage

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

Arguments

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

Value

none

Author(s)

Nicolas Delhomme

Examples

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

## End(Not run)
```

`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 `BamFile` 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`
- `genomeIntervals-readGff3`

Examples

```
# library
library(genomeIntervals)

# fetch the example data
gffFilePath <- fetchData("Dmel-mRNA-exon-r5.52.gff3.gz")
annot<-readGff3(gffFilePath,quiet=TRUE)
type(annot)
```

`getBamFileList` *Get a BamFileList from a list of filenames*

Description

A utility function to create a `BamFileList-class` 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,character'
getBamFileList(filenames = character(0), indexnames = character(0))
```

Arguments

`filenames`	a character vector containing fully defined BAM file filenames
`indexnames`	a character vector containing fully defined BAM index file filenames

Value

a `BamFileList-class`

See Also

`BamFileList-class` `dir`

Examples

```
# tutorial data - store the data in the BiocCache
tdir <- tutorialData()

  # creating a BamFileList using a directory and pattern
  # using filenames (from the Bioc cache)
  filenames <- dir(tdir,pattern="[A,C,T,G]{6}\\.bam$",full.names=TRUE)
  indexnames <- sapply(paste0(sub(".*_","",basename(filenames)),".bai"),fetchData)
  bfl <- getBamFileList(filenames,indexnames)

  # get them recursively
  filenames <- dir(path=tdir,pattern="[A,C,T,G]{6}\\.bam$",
                   full.names=TRUE,recursive=TRUE)
  indexnames <- sapply(paste0(sub(".*_","",basename(filenames)),".bai"),fetchData)
  bfl <- getBamFileList(filenames,indexnames)
```

`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.

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` and `stopCluster` in `makeCluster`

Examples

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

`print methods` *Pretty print the content of classes from the easyRNASeq package.*

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 `RNAseq`, `AnnotParam`, `BamParam` or `RnaSeqParam`
`verbose`	A logical to have a verbose or not output. Default to FALSE For object of the `RNAseq` class only.
`...`	Additional arguments, currently unused.

Value

Print information about the provided object.

Author(s)

Nicolas Delhomme

`RNAseq class` *Class "RNAseq"*

Description

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

Objects from the Class

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

Author(s)

Nicolas Delhomme

See Also

- `GRangesList`
- `RleList`
- `easyRNASeq`
- `easyRNASeq accessors`
- `easyRNASeq annotation`
- `easyRNASeq correction (FPKM)`
- `easyRNASeq coverage`
- `easyRNASeq summarization`
- `easyRNASeq print methods`

Examples

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

`RnaSeqParam class` *Class "RnaSeqParam"*

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 `summarizeOverlaps-methods` function is used to extract the read counts
- bpThe `easyRNASeq summarization functions` are used to extract the read covered bp 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-methods`
- `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

`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.

`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
`index.only`	simply return the index and not the barcode themselves.
`mc.cores`	A parameter ultimately passed to srdistance to enable parallel processing on mc.cores. On linux and Mac only, windows task remain serially processed.
`.name`	An internal string describing the filter
`show.barcode`	An integer specifying how many barcodes should be displayed in the final output.
`...`	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","TTGCGA")

invisible(download.file(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 `BamFileList` object
`param`	RnaSeqParam a `RnaSeqParam` 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
tdir <- tutorialData()
annot <- fetchData("Drosophila_melanogaster.BDGP5.77.with-chr.gtf.gz")

 # create the BamFileList, get the BAM and BAI index files from the Bioc cache
 filenames <- dir(tdir,pattern="[A,T].*\\.bam$",full.names=TRUE)
 indexnames <- sapply(paste0(sub(".*_","",basename(filenames)),".bai"),fetchData)
```

```
bamFiles <- getBamFileList(filenames,indexnames)

 # create the AnnotParam
 annotParam <- AnnotParam(annot,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 `BamFile` or `BamFileList` class
`header`	a boolean to (de)activate the check for a BAM header
`cross.validation`	a boolean - only valid for `BamFileList` 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

```
# retrieve the data
tdir <- tutorialData()

# get the bam file path from the Bioc cache
filenames <- dir(tdir,pattern="[A,C,T,G]{6}\\.bam$",full.names=TRUE)

# retrieve the index from the Bioc cache too
inxnames <- sapply(paste0(sub(".*_","",basename(filenames)),".bai"),fetchData)

bfl <-BamFileList(filenames,index=inxnames)

validate(bfl)
```

Index

- ∗ **classes**
  - AnnotParam class, 3
  - BamParam class, 4
  - RNAseq class, 44
  - RnaSeqParam class, 45
- ∗ **connection**
  - easyRNASeq annotation methods, 11
  - easyRNASeq island methods, 24
- ∗ **data**
  - easyRNASeq annotation methods, 11
  - easyRNASeq island methods, 24
  - easyRNASeq-datasets, 37
- ∗ **internal**
  - AnnotParam internal methods, 3
  - easyRNASeq annotation internal methods, 10
  - easyRNASeq internal methods, 22
  - easyRNASeq package, 25
  - easyRNASeq summarization internal methods, 29
  - easyRNASeq-global-variables, 37
- ∗ **manip**
  - easyRNASeq accessors, 9
  - easyRNASeq AnnotParam accessors, 12
  - easyRNASeq BamParam accessors, 14
  - easyRNASeq RnaSeqParam accessors, 27
- ∗ **methods**
  - basename methods, 5
  - BiocFileCache methods, 6
  - createSyntheticTranscripts,AnnotParamCharacter-method, 7
  - easyRNASeq annotation methods, 11
  - easyRNASeq correction methods, 16
  - easyRNASeq coverage methods, 18
  - easyRNASeq GenomicRanges package extension, 21
  - easyRNASeq island methods, 24
  - easyRNASeq summarization methods, 31
  - easyRNASeq,character-method, 33
  - edgeR additional methods, 38
  - file.exists methods, 39
  - IRanges additional methods, 42
  - parallel additional methods, 43
  - print methods, 44
  - ShortRead additional methods, 46
  - show methods, 49
  - simpleRNASeq,BamFileList,RnaSeqParam-method, 49
  - validate,BamFile-method, 51
- ∗ **package**
  - easyRNASeq package, 25
- .bestExonSummarization (easyRNASeq summarization internal methods), 29
- .catn (easyRNASeq internal methods), 22
- .checkArguments (easyRNASeq internal methods), 22
- .convertToUCSC (easyRNASeq internal methods), 22
- .doBasicCount (easyRNASeq summarization internal methods), 29
- .doCount (easyRNASeq summarization internal methods), 29
- .extendCountList (easyRNASeq summarization internal methods), 29
- .extractIRangesList (easyRNASeq internal methods), 22
- .geneModelAnnotation (easyRNASeq annotation internal methods), 10
- .geneModelSummarization, *32*
- .geneModelSummarization (easyRNASeq summarization internal methods), 29

.getArguments (easyRNASeq internal methods), 22
.getBmRange (easyRNASeq annotation internal methods), 10
.getGffRange (easyRNASeq annotation internal methods), 10
.getGtfRange (easyRNASeq annotation internal methods), 10
.getName (easyRNASeq internal methods), 22
.getWidth (easyRNASeq internal methods), 22
.get_cache (BiocFileCache methods), 6
.get_cache,ANY-method (BiocFileCache methods), 6
.list.files (easyRNASeq internal methods), 22
.list.files-defunct (easyRNASeq internal methods), 22
.normalizationDispatcher (easyRNASeq internal methods), 22
.onAttach, *38*
.onAttach (easyRNASeq-global-variables), 37
.readGffGtf (easyRNASeq annotation internal methods), 10
.validate (AnnotParam internal methods), 3

accessors (easyRNASeq accessors), 9
alignData (ShortRead additional methods), 46
AlignedRead, *46*, *47*
ANNOTATION.TYPE (easyRNASeq-global-variables), 37
annotations, *49*
AnnotParam, *3*, *5*, *8*, *10–14*, *28*, *44*, *46*, *49*, *50*
AnnotParam (easyRNASeq AnnotParam constructor), 13
annotParam (easyRNASeq RnaSeqParam accessors), 27
AnnotParam class, 3
AnnotParam internal methods, 3
AnnotParam,character-method (easyRNASeq AnnotParam constructor), 13
AnnotParam,GRanges-method (easyRNASeq AnnotParam constructor), 13
AnnotParam,missing-method (easyRNASeq AnnotParam constructor), 13
annotParam,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
AnnotParam-accessors (easyRNASeq AnnotParam accessors), 12
AnnotParam-class (AnnotParam class), 3
AnnotParamCharacter, *7*
AnnotParamCharacter-class (AnnotParam class), 3
AnnotParamObject-class (AnnotParam class), 3
assay (easyRNASeq package), 25

BamFile, *5*, *39*, *50–52*
BamFileList, *5*, *50–52*
BamFileList (easyRNASeq package), 25
BamFileList-class (easyRNASeq package), 25
BamParam, *4*, *14*, *15*, *28*, *44*, *46*, *49*, *50*
BamParam (easyRNASeq BamParam constructor), 15
bamParam (easyRNASeq RnaSeqParam accessors), 27
BamParam class, 4
BamParam,ANY-method (easyRNASeq BamParam constructor), 15
bamParam,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
BamParam-accessors (easyRNASeq BamParam accessors), 14
BamParam-class (BamParam class), 4
barcodePlot (ShortRead additional methods), 46
barcodePlot,AlignedRead-method (ShortRead additional methods), 46
barcodePlot,DNAStringSet-method (ShortRead additional methods), 46
barcodePlot,ShortReadQ-method (ShortRead additional methods), 46
basename (basename methods), 5
basename methods, 5

basename,BamFile-method (basename methods), 5
basename,BamFileList-method (basename methods), 5
BiocFileCache, *6*
BiocFileCache methods, 6
BiocParallel, *26*
biomaRt, *10*, *26*
Biostrings, *13*, *26*
BSgenome, *13*, *26*

chastityFilter (ShortRead additional methods), 46
chastityFilter,SRFilter-method (ShortRead additional methods), 46
chromosomeFilter (easyRNASeq package), 25
chrSize (easyRNASeq accessors), 9
chrSize,RNAseq-method (easyRNASeq accessors), 9
chrSize<- (easyRNASeq accessors), 9
chrSize<-,RNAseq,integer-method (easyRNASeq accessors), 9
chrSize<-,RNAseq,list-method (easyRNASeq accessors), 9
clusterApply, *43*
colnames (easyRNASeq GenomicRanges package extension), 21
colnames,GRanges-method (easyRNASeq GenomicRanges package extension), 21
colnames,GRangesList-method (easyRNASeq GenomicRanges package extension), 21
compose (easyRNASeq package), 25
countBy (easyRNASeq RnaSeqParam accessors), 27
countBy,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
createSyntheticTranscripts (createSyntheticTranscripts,AnnotParamCharacter-method), 7
createSyntheticTranscripts,AnnotParamCharacter-method, 7
createSyntheticTranscripts,character-method (createSyntheticTranscripts,AnnotParamCharacter-method), 7

DataFrame, *21*
datasource (easyRNASeq AnnotParam accessors), 12
datasource,AnnotParam-method (easyRNASeq AnnotParam accessors), 12
datasource,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
Defunct functions, 9
demultiplex (ShortRead additional methods), 46
demultiplex,AlignedRead-method (ShortRead additional methods), 46
demultiplex,DNAStringSet-method (ShortRead additional methods), 46
demultiplex,ShortReadQ-method (ShortRead additional methods), 46
DGEList, *39*
dir, *41*

easyRNASeq, *9*, *25*, *30*, *31*, *45*, *49*
easyRNASeq (Defunct functions), 9
easyRNASeq accessors, 9
easyRNASeq annotation internal methods, 10
easyRNASeq annotation methods, 11, *26*
easyRNASeq AnnotParam accessors, 12
easyRNASeq AnnotParam constructor, 13
easyRNASeq BamParam accessors, 14
easyRNASeq BamParam constructor, 15
easyRNASeq correction methods, 16, *26*
easyRNASeq coverage methods, 18, *26*
easyRNASeq defunct annotation methods, 20
easyRNASeq GenomicRanges package extension, 21
easyRNASeq internal methods, 22
easyRNASeq island methods, 24
easyRNASeq package, 25
easyRNASeq RnaSeqParam accessors, 27
easyRNASeq RnaSeqParam constructor, 28
easyRNASeq summarization internal methods, 29
easyRNASeq normalization methods, *26*, 31
easyRNASeq,character-method, 33

easyRNASeq,RNAseq-method (Defunct functions), 9
easyRNASeq-datasets, 37
easyRNASeq-defunct (easyRNASeq,character-method), 33
easyRNASeq-global-variables, 37
easyRNASeq-package (easyRNASeq package), 25
edgeR, *25*, *26*
edgeR additional methods, 38
edgeR methods, *26*
edgeR:DGEList, *36*
exonCounts (easyRNASeq summarization methods), 31
exonCounts,RNAseq-method (easyRNASeq summarization methods), 31

featureCounts (easyRNASeq summarization methods), 31
featureCounts,RNAseq-method (easyRNASeq summarization methods), 31
fetchAnnotation (Defunct functions), 9
fetchAnnotation-defunct (easyRNASeq defunct annotation methods), 20
fetchCoverage, *9*
fetchCoverage (Defunct functions), 9
fetchCoverage,RNAseq-method (Defunct functions), 9
fetchCoverage-deprecated (easyRNASeq coverage methods), 18
fetchData (BiocFileCache methods), 6
fetchData,character-method (BiocFileCache methods), 6
file.exists (file.exists methods), 39
file.exists methods, 39
file.exists,BamFile-method (file.exists methods), 39
fileName (easyRNASeq accessors), 9
fileName,RNAseq-method (easyRNASeq accessors), 9
fileName<- (easyRNASeq accessors), 9
fileName<-,RNAseq-method (easyRNASeq accessors), 9
findIslands, *32*
findIslands (easyRNASeq island methods), 24
findIslands,RNAseq-method (easyRNASeq island methods), 24

GAlignments, *21*
geneCounts (easyRNASeq summarization methods), 31
geneCounts,RNAseq-method (easyRNASeq summarization methods), 31
geneModel (easyRNASeq accessors), 9
geneModel,RNAseq-method (easyRNASeq accessors), 9
geneModel<- (easyRNASeq accessors), 9
geneModel<-,RNAseq-method (easyRNASeq accessors), 9
Genome_intervals, *8*, *40*
Genome_intervals_stranded, *10*
genomeIntervals, *23*, *26*
genomeIntervals additional methods, 40
genomeIntervals object, *40*
GenomicAlignments, *49*
genomicAnnotation (easyRNASeq accessors), 9
genomicAnnotation,RNAseq-method (easyRNASeq accessors), 9
genomicAnnotation<- (easyRNASeq accessors), 9
genomicAnnotation<-,RNAseq-method (easyRNASeq accessors), 9
GenomicFeatures, *26*
GenomicRanges, *21*, *23*, *26*
getAnnotation, *14*, *20*
getAnnotation (easyRNASeq annotation methods), 11
getAnnotation,AnnotParam-method (easyRNASeq annotation methods), 11
getBamFileList, 41, *50*
getBamFileList,character,character-method (getBamFileList), 41
getBamFileList,character,missing-method (getBamFileList), 41
getBM, *4*, *11*, *35*
GRanges, *8*, *10–14*, *21*
GRanges (easyRNASeq package), 25
GRanges-class (easyRNASeq package), 25
GRangesList, *21*, *34*, *45*
GTF.FIELDS (easyRNASeq-global-variables), 37

hist, *24*

intervals, *23*
IRanges, *26*
IRanges (easyRNASeq package), 25
IRanges additional methods, 42
IRangesList, *42*
islandCounts (easyRNASeq summarization methods), 31
islandCounts,RNAseq-method (easyRNASeq summarization methods), 31

knownOrganisms (Defunct functions), 9
knownOrganisms-defunct (easyRNASeq defunct annotation methods), 20

librarySize (easyRNASeq accessors), 9
librarySize,RNAseq-method (easyRNASeq accessors), 9
librarySize<- (easyRNASeq accessors), 9
librarySize<-,RNAseq-method (easyRNASeq accessors), 9
list.files, *35*
listDatasets, *11*

makeCluster, *43*

naPositionFilter (ShortRead additional methods), 46
naPositionFilter,SRFilter-method (ShortRead additional methods), 46
nFilter (easyRNASeq package), 25

optionally apply, *33*
organismName (Defunct functions), 9
organismName,RNAseq-method (Defunct functions), 9
organismName<- (Defunct functions), 9
organismName<-,RNAseq-method (Defunct functions), 9

paired (easyRNASeq BamParam accessors), 14
paired,BamParam-method (easyRNASeq BamParam accessors), 14
paired,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
parallel, *26*
parallel additional methods, 43
parallelize (parallel additional methods), 43
parallelize,BamFileList,function-method (parallel additional methods), 43
parallelize,GRangesList,function-method (parallel additional methods), 43
parallelize,list,function-method (parallel additional methods), 43
parallelize,vector,function-method (parallel additional methods), 43
plotNormalizationFactors (edgeR additional methods), 38
plotNormalizationFactors,DGEList,character,character-metho (edgeR additional methods), 38
precision (easyRNASeq RnaSeqParam accessors), 27
precision,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
print (print methods), 44
print methods, 44
print,AnnotParam-method (print methods), 44
print,BamParam-method (print methods), 44
print,RNAseq-method (print methods), 44
print,RnaSeqParam-method (print methods), 44

RangedSummarizedExperiment, *25*, *26*, *36*, *49*, *50*
RangedSummarizedExperiment-class (easyRNASeq package), 25
ranges (IRanges additional methods), 42
ranges,RNAseq-method (IRanges additional methods), 42
readCounts, *17*
readCounts (easyRNASeq accessors), 9
readCounts,RNAseq-method (easyRNASeq accessors), 9
readCounts<- (easyRNASeq accessors), 9
readCounts<-,RNAseq-method (easyRNASeq accessors), 9
readCoverage (easyRNASeq accessors), 9

readCoverage,RNAseq-method (easyRNASeq accessors), 9
readCoverage<- (easyRNASeq accessors), 9
readCoverage<-,RNAseq-method (easyRNASeq accessors), 9
readGffGtf, *11*, *35*
readIslands (easyRNASeq accessors), 9
readIslands,RNAseq-method (easyRNASeq accessors), 9
readIslands<- (easyRNASeq accessors), 9
readIslands<-,RNAseq-method (easyRNASeq accessors), 9
readLength (easyRNASeq accessors), 9
readLength,RNAseq-method (easyRNASeq accessors), 9
readLength<- (easyRNASeq accessors), 9
readLength<-,RNAseq-method (easyRNASeq accessors), 9
reduce (easyRNASeq internal methods), 22
reduce,RNAseq-method (easyRNASeq internal methods), 22
Rle, *20*
RleList, *45*
RNAseq, *17*, *19*, *26*, *30–32*, *36*, *42*, *44*, *49*
RNAseq (RNAseq class), 44
RNAseq class, 44
RNAseq-class (RNAseq class), 44
RnaSeqParam, *3*, *5*, *27*, *28*, *44*, *49*, *50*
RnaSeqParam (easyRNASeq RnaSeqParam constructor), 28
RnaSeqParam class, 45
RnaSeqParam,ANY-method (easyRNASeq RnaSeqParam constructor), 28
RnaSeqParam-accessors (easyRNASeq RnaSeqParam accessors), 27
RnaSeqParam-class (RnaSeqParam class), 45
RobinsonDelhomme2014 (easyRNASeq-datasets), 37
row_colnames, *21*
RPKM (easyRNASeq correction methods), 16
RPKM,matrix,ANY,vector,vector-method (easyRNASeq correction methods), 16
RPKM,RNAseq,ANY,ANY,ANY-method (easyRNASeq correction methods), 16
RPKM,RNAseq-method (easyRNASeq correction methods), 16
Rsamtools, *26*

seqnames,RNAseq-method (easyRNASeq accessors), 9
ShortRead, *26*
ShortRead additional methods, 46
ShortRead methods, *26*
ShortRead:readAligned, *20*, *31*, *36*
show methods, 49
show,AnnotParam-method (show methods), 49
show,BamParam-method (show methods), 49
show,RNAseq-method (show methods), 49
show,RnaSeqParam-method (show methods), 49
simpleRNASeq, *9*
simpleRNASeq (simpleRNASeq,BamFileList,RnaSeqParam-method), 49
simpleRNASeq,BamFileList,RnaSeqParam-method, 49
SRFilter, *47*
srFilter, *26*
SRFilterResult (easyRNASeq package), 25
strand (easyRNASeq internal methods), 22
strand,RNAseq-method (easyRNASeq internal methods), 22
strand<- (easyRNASeq internal methods), 22
strand<-,RNAseq,ANY-method (easyRNASeq internal methods), 22
strand<-,RNAseq-method (easyRNASeq internal methods), 22
stranded (easyRNASeq BamParam accessors), 14
stranded,BamParam-method (easyRNASeq BamParam accessors), 14
stranded,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27
strandProtocol (easyRNASeq BamParam accessors), 14
strandProtocol,BamParam-method (easyRNASeq BamParam accessors), 14
strandProtocol,RnaSeqParam-method (easyRNASeq RnaSeqParam accessors), 27

`SummarizedExperiment` *(easyRNASeq package)*, 25
`summarizes`, *49*

`transcriptCounts` *(easyRNASeq summarization methods)*, 31
`transcriptCounts,RNAseq-method` *(easyRNASeq summarization methods)*, 31
`TUTORIAL.DATA` *(easyRNASeq-global-variables)*, 37
`tutorialData` *(BiocFileCache methods)*, 6
`tutorialData,ANY-method` *(BiocFileCache methods)*, 6
`type` *(easyRNASeq package)*, 25
`type,AnnotParam-method` *(easyRNASeq AnnotParam accessors)*, 12
`type,Genome_intervals-method` *(genomeIntervals additional methods)*, 40

`unsafeAppend` *(easyRNASeq GenomicRanges package extension)*, 21
`unsafeAppend,GAlignments,GAlignments-method` *(easyRNASeq GenomicRanges package extension)*, 21

`validate` *(validate,BamFile-method)*, 51
`validate,BamFile-method`, 51
`validate,BamFileList-method` *(validate,BamFile-method)*, 51
`VIGNETTE.DATA` *(easyRNASeq-global-variables)*, 37
`vignetteData` *(BiocFileCache methods)*, 6
`vignetteData,ANY-method` *(BiocFileCache methods)*, 6

`yieldSize` *(easyRNASeq BamParam accessors)*, 14
`yieldSize,BamParam-method` *(easyRNASeq BamParam accessors)*, 14
`yieldSize,RnaSeqParam-method` *(easyRNASeq RnaSeqParam accessors)*, 27