Package 'DEXSeq'

October 27, 2015

Version 1.16.0

Title Inference of differential exon usage in RNA-Seq

Author Simon Anders <sanders@fs.tum.de> and Alejandro Reyes <alejandro.reyes@embl.de>, both at EMBL Heidelberg

Maintainer Alejandro Reyes <alejandro.reyes@embl.de>

Imports BiocGenerics, biomaRt, hwriter, methods, stringr, Rsamtools, statmod, geneplotter, genefilter, RColorBrewer

Depends BiocParallel, Biobase, IRanges (>= 2.1.10), GenomicRanges (>= 1.19.6), DESeq2 (>= 1.9.11)

Suggests GenomicFeatures (>= 1.13.29), pasilla (>= 0.2.22), parathyroidSE, BiocStyle, knitr

Enhances parallel

Description The package is focused on finding differential exon usage using RNA-seq exon counts between samples with different experimental designs. It provides functions that allows the user to make the necessary statistical tests based on a model that uses the negative binomial distribution to estimate the variance between biological replicates and generalized linear models for testing. The package also provides functions for the visualization and exploration of the results.

License GPL (>= 3)

URL

biocViews Sequencing, RNASeq, DifferentialExpression

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

`counts` *Accessors for the 'counts' slot of a DEXSeqResults object.*

Description

The counts slot holds the count data as a matrix of non-negative integer count values, one row for each observational unit (gene or the like), and one column for each sample.

Usage

```
## S4 method for signature 'DEXSeqResults'
counts(object,normalized=FALSE)
```

Arguments

`object`	a `DEXSeqResults` object.
`normalized`	logical indicating whether or not to divide the counts by the size factors or normalization factors before returning (normalization factors always preempt size factors)

Value

an integer matrix

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
head( counts( dxd ))
```

DEXSeq *Performs the differential exon usage test in a single command*

Description

This function is a wrapper that calls the necessary functions to perform a differential exon usage test in a single command.

Usage

```
DEXSeq(object,
  fullModel=design(object),
  reducedModel = ~ sample + exon,
  BPPARAM=MulticoreParam(workers=1),
  fitExpToVar="condition", quiet=TRUE )
```

Arguments

object	An DEXSeqDataSet object.
fullModel	The full model formula
reducedModel	Null model formula.
BPPARAM	A "BiocParallelParam" instance. See ?bplapply for details.
fitExpToVar	A variable name contained in the sample data. The expression values will be fitted to this variable using the the formula " ~ sample + fitExpToVar * exon".
quiet	Whether to print messages at each step

Value

A DEXSeqResults object.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxr <- DEXSeq( dxd )
```

DEXSeq-defunct *This functions are deprecated and will become defunct.*

Description

The ExonCountSet object has been deprecated and substituted by the DEXSeqDataSet object. Therefore, all the functions and methods for the ExonCountSet object were deprecated. These functions have been subsituted by new functions and methods designed for DEXSeqDataSet objects.

Details

The replacements of defunct functions are summarized in the following items.

- newExonCountSet: `DEXseqDataSet`
- DEUresultTable: `DEXseqResults`
- subsetByGenes:
- geneCountTable:
- estimateExonDispersionsForModelFrame_BM: estimateDispersions-DEXSeqDataSet
- estimateDispersions_BM: estimateDispersions-DEXSeqDataSet
- testGeneForDEU_BM: `testForDEU`
- testForDEU_BM: `testForDEU`
- doCompleteDEUAnalysis: `DEXSeq`
- makeCompleteDEUAnalysis_BM: `DEXSeq`
- read.HTSeqCounts: `DEXSeqDataSetFromHTSeq`
- countTableForGene:
- fitDispersionFunction: estimateDispersions-DEXSeqDataSet
- estimatelog2FoldChange: `estimateExonFoldChanges`
- modelFrameForGene:
- buildExonCountSet: `DEXSeqDataSetFromSE`
- constructModelFrame:
- getCountVector:
- estimateExonDispersion: estimateDispersions-DEXSeqDataSet
- testExonForDEU: `testForDEU`
- doCompleteDEUAnalysis: `DEXSeq`

DEXSeqDataSet-class *DEXSeqDataSet object and constructors*

Description

The `DEXSeqDataSet` is a subclass of `DESeqDataSet`, specifically designed to adapt the `DESeqDataSet` to test for differences in exon usage.

Usage

```
DEXSeqDataSet( countData, sampleData,
  design= ~ sample + exon + condition:exon,
  featureID, groupID, featureRanges=NULL,
  transcripts=NULL, alternativeCountData=NULL)

DEXSeqDataSetFromHTSeq(
  countfiles, sampleData,
  design= ~ sample + exon + condition:exon,
  flattenedfile=NULL )

DEXSeqDataSetFromSE( SE,
  design= ~ sample + exon + condition:exon )
```

Arguments

countData	A matrix of count data of non-negative integer values. The rows correspond to counts for each exon counting bin, the columns correspond to samples. Note that biological replicates should each get their own column, while the counts of technical replicates (i.e., several sequencing runs/lanes from the same sample) should be summed up into a single column
alternativeCountData	DEXSeq can be also used for test for differences in exon inclusion based on the number of reads supporting the inclusion of an exon and the number of reads supporting the exclusion of an exon. A matrix of count data of non-negative integer values.The rows correspond to exonic regions and the columns correspond to samples. This matrix should contain the number of exon-exon junction reads that skip each exon in each sample. If NULL, then the sum of the other exons belonging to the same gene is considered for testing (i.e. the normal DEXSeq approach).
countfiles	A character vector containing the path to the files that were originated with the script 'dexseq_count.py'.
sampleData	A `data.frame` with the annotation (e.g. treatments, or tissue types, or phenotypes, or the like). The number of rows in the data frame must to be equal to the number of columns of the countData matrix, assigning the annotation of each sample.
design	A formula which specifies the design of the experiment. It must specify an interaction term between a variable from the sampleData columns with the 'exon' variable. By default, the design will be '~ sample + exon + condition:exon'. This formula indicates the contrast between 'condition' and exon', i.e. differences in exon usage due to changes in the 'condition' variable. See the vignette for more examples of other designs.
featureID	A character vector of counting regions identifiers ordered according to the rows in countData. The identifiers names can be repeated between groups but not within groups.
groupID	A vector of group identifiers ordered according to its respective row in countData. It must reflect the sets of counting regions belonging to the same group, for example, exon bins in belonging to the same gene should have the same group identifier.
featureRanges	Optional. `GRanges` or `GRangesList` describing the genomic coordinates of each of the rows of countData.
transcripts	Optional. A `list` of the same length as the number of columns of countData. Each element of the list should contain the transcript identifiers of the
flattenedfile	A character vector containing the path to the flattened annotation file that was originated with the script 'dexseq_prepare_annotation.py'.
SE	A `SummarizedExperiments` object, originated using the function `SummarizeOverlaps`. See examples for more details.

Value

A DEXSeqDataSet object.

See Also

`DEXSeqDataSetFromHTSeq` `DEXSeqDataSetFromSE`

Examples

```
## Not run:
  ########################################
  ### From the output of the          ###
  ### acconpaning python scripts      ###
  ########################################

  inDir = system.file("extdata", package="pasilla", mustWork=TRUE)
  flattenedfile = file.path(inDir, "Dmel.BDGP5.25.62.DEXSeq.chr.gff")
  sampleData = data.frame(
      condition = c( rep("treated", 3), rep("untreated", 4) ),
      type = c("single", "paired", "paired", "single", "single", "paired", "paired") )

  countFiles <- list.files(inDir, pattern="fb.txt")
  rownames( sampleData ) <- countFiles

  DEXSeqDataSetFromHTSeq(
    countfiles=file.path( inDir, countFiles ),
    sampleData = sampleData,
    design = ~ sample + exon + type:exon + condition:exon,
    flattenedfile=flattenedfile )


  ########################################
  ### From GRanges derived objects    ###
  ########################################

  library(GenomicRanges)
  library(GenomicFeatures)
  library(GenomicAlignments)

  hse <- makeTxDbFromBiomart( biomart="ensembl",
                              dataset="hsapiens_gene_ensembl" )

  bamDir <- system.file(
    "extdata", package="parathyroidSE", mustWork=TRUE )
  fls <- list.files( bamDir, pattern="bam$", full=TRUE )

  bamlst <- BamFileList(
    fls, index=character(),
    yieldSize=100000, obeyQname=TRUE )

  exonicParts <- disjointExons( hse, aggregateGenes=FALSE )

  SE <- summarizeOverlaps( exonicParts, bamlst,
    mode="Union", singleEnd=FALSE,
    ignore.strand=TRUE, inter.feature=FALSE, fragments=TRUE )

  colData(SE)$condition <- c("A", "A", "B")

  DEXSeqDataSetFromSE( SE,
    design= ~ sample + exon + condition:exon )
```

```
  ######################################
  ### From elementary data structures ###
  ######################################
  countData <- matrix( rpois(10000, 100), nrow=1000 )
  sampleData <- data.frame(
      condition=rep( c("untreated", "treated"), each=5 ) )
  design <- formula( ~ sample + exon + condition:exon )
  groupID <- rep(
      paste0("gene", 1:10),
      each= 100 )
  featureID <- rep(
      paste0("exon", 1:100),
      times= 10 )
  DEXSeqDataSet( countData, sampleData, design,
              featureID, groupID )


## End(Not run)
```

DEXSeqHTML *DEXSeq HTML report writer*

Description

This function generates an HTML report from the results of a DEXSeqResults object. Gives an simple report to explore differential exon usage results.

Usage

```
DEXSeqHTML(object, genes=NULL, path="DEXSeqReport", file="testForDEU.html",
  fitExpToVar="condition", FDR=0.1, color=NULL, color.samples=NULL,
 mart="", filter="", attributes="", extraCols=NULL, BPPARAM=MulticoreParam(workers=1))
```

Arguments

object	An DEXSeqResults object
genes	A character vector of gene identifiers to be included in the report. If NULL, the genes included in the report will be the significant hits at the given false discovery rate. See "FDR" below.
path	A path in the system where to write the report.
file	The name of the html file.
fitExpToVar	A variable contained in the design of the ecs; the counts will be fitted to this variable to get the plotting values.
FDR	A false discovery rate
color	A vector of colors, one for each of the levels of the values of "fitExpToVar".
color.samples	A vector of colors for each of the samples. If NULL, the colors of each sample will be asigned according to its corresponding condition. Useful to visualize complex experimental designs.
mart	object of class Mart, created with the useMart function, with dataset specified

filter	Filters (ONLY ONE) that should be used in the query. A possible list of filters can be retrieved using the function listFilters. Please note that the value of this filter will always be the geneIDs in the DEXSeqResults object.
attributes	Attributes you want to retrieve. A possible list of attributes can be retrieved using the biomaRt function listAttributes.
extraCols	A data frame with one or more columns to add to the report. For example, additional information about the genes. The data frame should be indexed by the gene names of the ExonCountSet object, e.g. the rownames of the data frame should correspond to the gene names.
BPPARAM	A 'BiocParallelParam' instance. See ?bplapply for details.

Value

This function will write an HTML report in the directory specified by 'path'. There, it will create an html file with the initial report page and a directory called "files" in which SVG files with the plots and other html files are placed. Different plots with different labels are generated for each gene: - counts: the raw data, for each sample - fitted expression: the fitted coefficients per compared condition (e.g.: treated, untreated) - fitted splicing: as 'expression', but after removing overall gene-level differential expression: this is the view most relevant for the interpretation of DEXSeq results, which are about changes in relative exon usage (i.e.: relative to overall gene expression)

Examples

```
## Not run:
  data(pasillaDEXSeqDataSet, package="pasilla")
  dxr <- DEXSeq( dxd )
  DEXSeqHTML( object=dxr )

## End(Not run)
```

DEXSeqResults-class *DEXSeqResults object*

Description

The DEXSeqResults object is a subclass of a DataFrame. It collects relevant information from a DEXSeqDataSet object with the results generated from testing for differences in exon usage.

Usage

```
DEXSeqResults( object )
```

Arguments

object	A DEXSeqDataSet object.

Value

A DEXSeqResults object.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
dxd <- testForDEU( dxd )
dxr <- DEXSeqResults( dxd )
```

`estimateDispersions` *Estimate the dispersions for a DEXSeqDataSet*

Description

This function obtains dispersion estimates for negative binomial distributed data for the specific case for DEXSeq.

Usage

```
## S4 method for signature 'DEXSeqDataSet'
estimateDispersions( object, fitType=c("parametric","local","mean"), maxit=100, niter=10, quiet=F
```

Arguments

`object`	A DEXSeqDataSet
`fitType`	Either "parametric", "local", or "mean" for the type of fitting of dispersions to the mean intensity. See ?estimateDispersions,DESeqDataSet-method for details.
`maxit`	Control parameter: maximum number of iterations to allow for convergence
`niter`	Number of times to iterate between estimation of means and estimation of dispersion.
`quiet`	Whether to print messages at each step
`formula`	Formula used to fit the dispersion estimates
`BPPARAM`	A "BiocParallelParam" instance. See `?bplapply` for details.

Details

See ?estimateDispersions,DESeqDataSet-method for details.

Value

A DEXSeqDataSet with the dispersion information filled in as metadata columns.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
```

`estimateExonFoldChanges` *Estimates exon usage coefficients from the fitted terms of the GLM.*

Description

This function calculates the exon usage coefficients and fold changes (on log2 scale) between the different conditions.

Usage

```
estimateExonFoldChanges( object,
   fitExpToVar = "condition", denominator = "",
   BPPARAM=MulticoreParam(workers=1), maxRowsMF=3000)
```

Arguments

`object`	An DEXSeqDataSet object.
`fitExpToVar`	A variable contained in `design(ecs)`. The expression values will be fitted to this variable using the the formula " ~ sample + fitExpToVar * exon".
`denominator`	A value of the sample annotation (e.g. condition) to use as a denominator in the log2 fold change. As a default, the function will take the annotation of the first sample.
`BPPARAM`	A "BiocParallelParam" instance. See `?bplapply` for details.
`maxRowsMF`	For the fold change estimation, the size of the model matrix for the fitted glm increases with the number of samples and the number of exons for a specific gene (see the DEXSeq paper for details). Since the glm fit for big models can be slow, the maxRowsMF parameter allows to set a threshold on the number of rows from the model frame (this will be number of samples x number of exons from a gene). For the genes passing this threshold the exon fold changes won't be calculated.

Details

Exon usage coefficients are calculated by fitting a GLM from the joint data of all exons of the same gene. The model frame can be found in the slot object@modelFrameBM of a `DEXSeqDataSet` object. The model '~ sample + fitExpToVar * exon' is fitted. The resulted coefficients are arranged and reformatted in order to remove gene expression effects (absorbed by the 'sample' variable in the formula), leaving only exon usage effects for each individual exon in each level of 'fitExpToVar'. These values are used by the function `plotDEXSeq`.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
dxd <- testForDEU( dxd )
dxd <- estimateExonFoldChanges( dxd )
```

`estimateSizeFactors` *Estimate the size factors for a DEXSeqDataSet*

Description

Estimate the size factors for a DEXSeqDataSet

Usage

```
  ## S4 method for signature 'DEXSeqDataSet'
estimateSizeFactors(object,locfunc=median,geoMeans)
```

Arguments

`object`	a DEXSeqDataSet
`locfunc`	a function to compute a location for a sample. By default, the median is used. However, especially for low counts, the `shorth` function from the genefilter package may give better results.
`geoMeans`	by default this is not provided and the geometric means of the counts are calculated within the function. A vector of geometric means from another count matrix can be provided for a "frozen" size factor calculation

Details

This function calls the method estimateSizeFactors for the DESeqDataSet object, and adapts it to the specific use to the DEXSeqDataSet.

Value

The DEXSeqDataSet passed as parameters, with the size factors filled in.

References

- Simon Anders, Wolfgang Huber: Differential expression analysis for sequence count data. Genome Biology 11 (2010) R106, http://dx.doi.org/10.1186/gb-2010-11-10-r106

`featureCounts` *Accessor functions for DEXSeqDataSet details*

Description

Accessor functions of the DEXSeqDataSet object.

Usage

```
featureCounts(object, normalized=FALSE)

featureIDs( object )
featureIDs( object ) <- value

exonIDs( object )
exonIDs( object ) <- value

groupIDs( object )
groupIDs( object ) <- value

geneIDs( object )
geneIDs( object ) <- value

sampleAnnotation( object )
```

Arguments

object	An DEXSeqDataSet object.
value	A character vector to replace previous values.
normalized	Logical indicating whether or not to divide the counts by the size factors or normalization factors before returning (normalization factors always preempt size factors)

Value

'featureCounts' access the counts per exonic region or feature region names. 'featureIDs' and 'exonIDs' are accessor functions for the exon bin or features identifiers. 'groupIDs' and 'geneIDs' are accessor functions for the character vector grouping the features, for example exonIDs from the same gene are grouped together by having the same value in the geneIDs. 'sampleAnnotation' is an accessor for the information from the samples.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
head( featureCounts( dxd ) )
head( featureIDs(dxd))
head( exonIDs(dxd))
```

---

findOverlaps *Methods from GRanges overlaps for the DEXSeq object*

Description

This function generates an MA plot.

Usage

```
  ## S4 method for signature 'DEXSeqResults,GenomicRanges'
subsetByOverlaps( query, subject, maxgap = 0L,
    minoverlap = 1L, type = c("any", "start", "end", "within", "equal"),
    algorithm = c("nclist", "intervaltree"), ignore.strand = FALSE )

  ## S4 method for signature 'DEXSeqResults,GenomicRanges'
findOverlaps( query, subject, maxgap = 0L,
    minoverlap = 1L, type = c("any", "start", "end", "within", "equal"),
    algorithm = c("nclist", "intervaltree"), ignore.strand = FALSE )
```

Arguments

`query`	Either a `DEXSeqResults` object.
`subject`	A GRanges or GRangesList object.
`maxgap, minoverlap, type, algorithm`	See `findOverlaps` in the IRanges package for a description of these arguments.
`ignore.strand`	See `findOverlaps` in the GenomicRanges package for a description of this.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
dxd <- testForDEU( dxd )
dxr <- DEXSeqResults( dxd )

interestingRegion = GRanges("chr2L", IRanges(start=3872658, end=3875302))

subsetByOverlaps( query=dxr, subject=interestingRegion )
findOverlaps( query=dxr, subject=interestingRegion )
```

---

perGeneQValue *Summarize per-exon p-values into per-gene q-values.*

Description

The use case for this function is the following analysis: given per-exon p-values for null hypothesis H0, we can determine the number of genes in which at least for one exon H0 is rejected. What is the associated false disovery rate?

Usage

```
perGeneQValue(object, p = "pvalue", method = perGeneQValueExact)
```

Arguments

object	An DEXSeqResults object.
p	A character string indicating the name of the columns in DEXSeqResults from which to take the per-exon p-values.
method	Use the default value. This is for debugging only.

Details

Details

Value

A named numeric vector, values are per-gene q-values, names are gene.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
dxd <- testForDEU( dxd )
dxr <- DEXSeqResults( dxd )

perGeneQValue(dxr)
```

---

plotDEXSeq *Visualization of the per gene DEXSeq results.*

Description

The function provides a plot to visualize read count data, expression and exon usage coefficients estimated from fitting a GLM model 'counts ~ fitExpToVar * exon'. The model frame used can be found in object@modelFrameBM. One GLM is fitted for each gene.

Usage

```
plotDEXSeq(object, geneID, FDR=0.1, fitExpToVar="condition",
  norCounts=FALSE, expression=TRUE, splicing=FALSE,
  displayTranscripts=FALSE, names=FALSE, legend=FALSE,
  color=NULL, color.samples=NULL, ...)
```

Arguments

object	A DEXSeqResults object. A DEXSeqDataSet object is also accepted, but only with the possibility of plotting the normalized read counts for each exon bin.
geneID	Gene identifier to visualize.
FDR	A false discovery rate.
fitExpToVar	A variable contained in the sample annotation of the DEXSeqDataSet, the expression values will be fitted to this variable using the formula count ~ fitExpToVar * exon.

norCounts	If TRUE, provides a plot of the counts normalized by the size factors.
expression	If TRUE, the function plots the fitted EXPRESSION estimates from the glm regression.
splicing	If TRUE, the samples gene expression effects are averaged out, leaving only exon usage coefficients.
displayTranscripts	If TRUE, the transcripts are displayed in the plot.
names	If TRUE, the names of the transcripts are shown.
legend	If TRUE, a legend is displayed.
color	A vector of colors for each of the levels of the factor in the design of the ExonCountSet object indicated by "fitExpToVar".
color.samples	A vector of colors for each of the samples. If NULL, the colors of each sample will be assigned according to its corresponding level from "fitExpToVar". This option is useful to visualize complex experimental designs.
...	Further graphical parameters (see par).

See Also

graphics, segments

Examples

```
## Not run:
  data(pasillaDEXSeqDataSet, package="pasilla")
  dxd <- estimateSizeFactors( dxd )
  dxd <- estimateDispersions( dxd )
  dxd <- testForDEU( dxd )
  dxr <- DEXSeqResults( dxd )

  plotDEXSeq( dxr, "FBgn0010909" )

## End(Not run)
```

---

plotDispEsts *Plot dispersion estimates*

Description

A simple helper function that plots the per-gene dispersion estimates together with the fitted mean-dispersion relationship. Internally, it is a wrapper for the pltoDispEsts method from DESeq2.

Usage

```
## S4 method for signature 'DEXSeqDataSet'
plotDispEsts(object, ymin,
  genecol = "black", fitcol = "red", finalcol = "dodgerblue",
  legend=TRUE, xlab, ylab, log = "xy", cex = 0.45, ...)
```

Arguments

object	a DESeqDataSet
ymin	the lower bound for points on the plot, points beyond this are drawn as triangles at ymin
genecol	the color for gene-wise dispersion estimates
fitcol	the color of the fitted estimates
finalcol	the color of the final estimates used for testing
legend	logical, whether to draw a legend
xlab	xlab
ylab	ylab
log	log
cex	cex
...	further arguments to plot

---

plotMA-methods *Generate an MA plot*

Description

This function generates an MA plot.

Usage

```
  ## S4 method for signature 'DEXSeqDataSet'
plotMA( object, alpha=0.1, ylim=c(-2, 2), foldChangeColumn=NULL, ... )

  ## S4 method for signature 'DEXSeqResults'
plotMA( object, alpha=0.1, ylim=c(-2, 2), foldChangeColumn=NULL, ... )
```

Arguments

object	Either a DEXSeqDataSeq or a DEXSeqResults.
alpha	the false discovery rate, i.e., threshold to the adjusted p values, to be used to colour the dots
ylim	y-limits of the plot
foldChangeColumn	Name of the column containing the fold changes to plot. If NULL, the first fold change column from the object will be taken.
...	Further parameters to be passed through to plot.

Examples

```
## Not run:
   data(pasillaDEXSeqDataSet, package="pasilla")
   dxd <- estimateSizeFactors( dxd )
   dxd <- estimateDispersions( dxd )
   dxd <- testForDEU( dxd )
   dxd <- estimateExonFoldChanges( dxd )
   dxr <- DEXSeqResults( dxd )

   plotMA( dxr )

   plotMA( dxd )


## End(Not run)
```

`testForDEU` *Test for Differential Exon Usage.*

Description

This will perform a likelihood ratio test for differential exon usage. Internally, it calls the DESeq2 function `nbinomLRT`.

Usage

```
testForDEU( object,
   fullModel = design(object),
   reducedModel = ~ sample + exon,
   BPPARAM=MulticoreParam(workers=1) )
```

Arguments

`object`	A DEXSeqDataSet object.
`fullModel`	The full model formula
`reducedModel`	Null model formula.
`BPPARAM`	A "BiocParallelParam" instance. See `?bplapply` for details.

Details

The information of the variables of the formulas should be present in the `colData` of the `DEXSeqDataSet` object.

Value

A `DEXSeqDataSet` with slots filled with information about the test.

Examples

```
data(pasillaDEXSeqDataSet, package="pasilla")
dxd <- estimateSizeFactors( dxd )
dxd <- estimateDispersions( dxd )
dxd <- testForDEU( dxd )
```

Index