# Package 'DEXSeq'

April 5, 2014

**Version** 1.8.0

**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 (>= 0.7.5), biomaRt, hwriter, methods, stringr,GenomicRanges, Rsamtools, statmod (>= 1.4.15)

**Depends** Biobase (>= 2.13.11)

**Suggests** GenomicFeatures (>= 1.13.29), pasilla (>= 0.2.13),parathyroidSE, BiocStyle

**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** HighThroughputSequencing, RNAseq, DifferentialExpression

## R topics documented:

---

`buildExonCountSet` *Makes an ExonCountSet object from R objects.*

---

## Description

This function inputs an GRanges object and a summarizedExperiment object and builds an ExonCountSet object.

## Usage

```
buildExonCountSet( summarizedExperiment, design, exonicParts )
```

### Arguments

| | |
|---|---|
| summarizedExperiment | A summarizedExperiments object, output of the function countReadsForDEXSeq. |
| design | A factor or data frame with the design annotation (e.g. treatments, or tissue types, or phenotypes, or the like). The length of the factor (or rows in the data frame) has to be equal to the number of columns of the assay data, assigning a condition to each sample. If it is a data frame, all the columns of the design need to be factors. |
| exonicParts | A GRanges object generated by the function prepareAnnotationForDEXSeq. |

### Value

An ExonCountSet object.

### Author(s)

From code kindly provided by Mike Love.

### Examples

```
## Not run:
   library(GenomicFeatures)
   library(Rsamtools)
  hse <- makeTranscriptDbFromBiomart(biomart="ensembl", dataset="hsapiens_gene_ensembl")
   exonicParts <- disjointExons( hse )

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

   ecs <- buildExonCountSet( SE, c("A", "A", "B"), exonicParts )

## End(Not run)
```

---

## constructModelFrame *Returns the model frame of an exonCountSet object for the GLM fits.*

---

### Description

Creates a data frame containing the model frame for a gene with the columns sample, exon, size factors, their respective counts and the design annotation.

### Usage

```
constructModelFrame(ecs)
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |

### Details

This function is called internally by several DEXSeq function, but is exposed as it might occasionally be usefule to users, too.

### Value

A data frame containing the model frame for a gene.

### Examples

```
data("pasillaExons", package="pasilla")
constructModelFrame(pasillaExons)
```

---

## countReadsForDEXSeq *Prepare annotation transcriptDb object for DEXSeq.*

---

### Description

WARNING: This function is deprecated, use summarizedOverlaps from the package GenomicRanges instead.

### Usage

```
countReadsForDEXSeq( exonicParts, bamFileList, scanBamParam =
                 ScanBamParam(), singleEnd = TRUE, ignoreStrand = TRUE,
                 mode = function(features, reads, ignore.strand,
                 inter.feature = FALSE) { countOverlaps(features,
                 reads, ignore.strand = ignoreStrand)})
```

### Arguments

| | |
|---|---|
| exonicParts | An GRanges object. |
| bamFileList | A BamFileList object. |
| scanBamParam | Function ScanBamParam to create a parameter object influencing what fields and which records are imported from a BAM file. |
| singleEnd | Logical. Indicating whether the reads are single-end or paired-end. |
| ignoreStrand | Logical. Indicating whether the strand of the reads should be ignored. Useful for data generated by strand-specific protocols. |
| mode | A function with the method used to count the overlaps to exons. The default allows a read fragment to be counted in two exons, if it overlaps with both of them. |

## Value

A GRanges object.

## Author(s)

From code kindly provided by Mike Love.

## Examples

```
## Not run:
   library(GenomicFeatures)
  hse <- makeTranscriptDbFromBiomart(biomart="ensembl", dataset="hsapiens_gene_ensembl")
   exonicParts <- prepareAnnotationForDEXSeq( hse )

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

   SE <- countReadsForDEXSeq( exonicParts, bamlst )

## End(Not run)
```

---

# counts *Accessors for the 'counts' slot of a ExonCountSet object.*

---

## Description

The counts slot holds the count data as a matrix of non-negative integer count values, one row for each observational unit (a counting bin, i.e., an exon or part of an exon), and one column for each sample.

## Usage

```
## S4 method for signature ExonCountSet
counts(object, normalized=FALSE)
## S4 replacement method for signature ExonCountSet,matrix
counts(object) <- value
```

## Arguments

| | |
|---|---|
| `object` | An ExonCountSet object. |
| `normalized` | If TRUE, the counts will be normalized by the size factors. |
| `value` | An integer matrix of counts, each row corresponding to an exon and each column corresponding to a sample. |

## Examples

```
   data("pasillaExons", package="pasilla")
   head( counts( pasillaExons ) )
```

countTableForGene *Count table for a given geneID.*

## Description

This function returns a matrix of non negative integers containing a count table for a specified geneID from an ExonCountSet object. The count table contains one row for every counting bin of the gene and a column for every sample.

## Usage

```
countTableForGene(ecs, geneID, normalized=FALSE, withDispersion=FALSE)
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet. |
| geneID | A geneID to get the count table. |
| normalized | If TRUE, the raw counts will be normalized by the size factors. |
| withDispersion | If TRUE, an extra column with the dispersion estimate used in the test will added to the count table. |

## See Also

estimateSizeFactors

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
countTableForGene(pasillaExons, "FBgn0085442", normalized=FALSE)
```

---

design *Accessor function for the design annotation from a ExonCountSet object.*

## Description

The design vector is a factor or data frame that assigns to each column of the count data a condition (or treatment, or phenotype, or the like). This information is stored in the ExonCountSet's "phenoData" slot as a row.

## Usage

```
## S4 method for signature ExonCountSet
design(object, drop=TRUE, asAnnotatedDataFrame=FALSE)
## S4 replacement method for signature ExonCountSet
design(object) <- value
```

### Arguments

| | |
|---|---|
| object | An ExonCountSet |
| drop | Indicates whether to return a single factor instead of a data frame in case of a one-way design |
| asAnnotatedDataFrame | Indicates whether the result should be presented as an AnnotatedDataFrame. |
| value | A vector or matrix with conditions for the samples, one row for each column in the count data. |

### Author(s)

Simon Anders, sanders@fs.tum.de

### Examples

```
library(DEXSeq)
data("pasillaExons", package="pasilla")
design( pasillaExons )
```

---

## DEUresultTable *Get a result table from the analysis workflow.*

---

### Description

This function returns a data frame with the summary of the results from the analysis workflow. It accesses the fData slots with information of the dispersion estimates obtained from the function fitDispersionFunction, the p values, and adjusted p values obtained from the function testForDEU, and log2 fold changes obtained from the function estimatelog2FoldChanges.

### Usage

```
DEUresultTable(ecs)
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |

### Value

A data frame with a summary of the analysis workflow.

## Examples

```
## Not run:
   data("pasillaExons", package="pasilla")
   pasillaExons <- estimateSizeFactors( pasillaExons )
   pasillaExons <- estimateDispersions( pasillaExons )
   pasillaExons <- fitDispersionFunction( pasillaExons )
   pasillaExons <- testForDEU( pasillaExons )
   res <- DEUresultTable( pasillaExons )

## End(Not run)
```

---

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

---

## Description

The function prepareAnnotationForDEXSeq was replaced by the function disjointExons from the package GenomicFeatures.

---

DEXSeqHTML *HTML report writer*

---

## Description

This function generates an HTML report from the results from testForDEU saved in an ExonCountSet object. It uses the information from the function DEUresultTable and plotting from plotDEXSeq. This gives an easy way of exploring the results of the tests.

## Usage

```
DEXSeqHTML(ecs, geneIDs=NULL, path="DEXSeqReport", file="testForDEU.html",
     fitExpToVar="condition", FDR=0.1, color=NULL, color.samples=NULL,
     mart="", filter="", attributes="", extraCols=NULL, nCores=1)
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object |
| geneIDs | A character vector of gene identificators to be included in the report. If left 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. (See plotDEXSeq for details. |

| | |
|---|---|
| FDR | A false discovery rate for the result. |
| 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 ExonCountSet 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. |
| nCores | Number of cores to be used. The parallel package must be loaded in order to spread the job onto several cores. |

## 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)

To see an example please visit http://www-huber.embl.de/pub/DEXSeq/psfb/testForDEU.html.

## See Also

hwrite

## Examples

```
## Not run:
          data("pasillaExons", package="pasilla")
          pasillaExons <- estimateSizeFactors( pasillaExons )
          pasillaExons <- estimateDispersions( pasillaExons )
          pasillaExons <- fitDispersionFunction( pasillaExons )
          pasillaExons <- testForDEU( pasillaExons )
          DEXSeqHTML( pasillaExons )

## End(Not run)
```

`doCompleteDEUAnalysis` *Perform complete differential exon usage analysis*

## Description

This function performs a complete differential exon usage analysis, calling all the necessary functions and returning an `ExonCountSet` object with p values, adjusted p values and fold change estimates.

## Usage

```
doCompleteDEUAnalysis( ecs,
     formula0 = ~ sample + exon,
     formula1 = ~ sample + exon + condition:exon,
     minCount = 10,
     nCores = 1,
     path = NULL,
     FDR = 0.1,
     fitExpToVar = "condition",
     color = NULL,
     color.samples = NULL )
```

## Arguments

| | |
|---|---|
| `ecs` | An ExonCountSet object. |
| `formula0` | Formula for the reduced (null) model, to be passed to `testForDEU`; see there for details. |
| `formula1` | Formula for the full model, to be passed to `estimateDispersions` and `testForDEU`; see there for details. |
| `minCount` | Exons with less than 'minCount' reads (summed over all samples) are excluded from the test. See `estimateDispersions` for details. |
| `nCores` | Number of CPU cores to be used when running `estimateDispersions` and `testForDEU`. Load the "parallel" package beforehands if you want to use more than one core. |
| `path` | A file system path to the directory into which the HTML report generated by `DEXSeqHTML` shoudl be written. If NULL, no report will be created. |
| `FDR` | Argument passed on to `DEXSeqHTML`; see there for details. |
| `fitExpToVar` | Argument passed on to `DEXSeqHTML`; see there for details. |
| `color` | Argument passed on to `DEXSeqHTML`; see there for details. |
| `color.samples` | Argument passed on to `DEXSeqHTML`; see there for details. |

## Value

An object of class ExonCountSet.

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- doCompleteDEUAnalysis( pasillaExons,
   formula0 = ~ sample + type * exon,
   formula1 = ~ sample + type * exon + condition * exon )
```

---

`estimateDispersions` *Estimate dispersions*

---

## Description

This function estimates for each counting bin of the `ExonCountSet` `object` a dispersion value. It stores these values in `fData(ecs)$dispBeforeSharing`.

## Usage

```
## S4 method for signature ExonCountSet
estimateDispersions( object,
         formula = ~ sample + exon + condition : exon,
         minCount = 10, nCores = 1 )
```

## Arguments

| | |
|---|---|
| `object` | An ExonCountSet object. |
| `formula` | Formula used in the GLM to estimate the dispersion values. The terms in the formula must be design columns of the ExonCountSet object, the l.h.s. will be the counts for each exon. |
| `minCount` | Counting bins with less than `minCount` counts (summed over all samples) are skipped in the tests. This reduces computation time, as counting bins with very few counts cannot give a significant signal anyway. For skipped counting bins, the `testable` column in `fData` is set to `FALSE`. |
| `nCores` | Number of cores to be used to estimate the dispersions. The `parallel` package must be loaded in order to spread the job onto several cores. |

## Details

For the dispersion estimation, we use the Cox-Reid conditional maximum likelihood method of McCarthy et al. (Nucl Acid Res., 2012, 40:4288), which they devised for the `edgeR` package.

## Value

An object of class `ExonCountSet` with dispersion `featureData(object)$dispersion_CR_est)` parameters filled).

## Examples

```
if(suppressWarnings(require("pasilla", quietly=TRUE, character.only=TRUE))){

   data("pasillaExons", package="pasilla")
   pasillaExons <- estimateSizeFactors( pasillaExons )
   pasillaExons <- estimateDispersions( pasillaExons )
   head( fData(pasillaExons)$dispBeforeSharing )

}
```

---

`estimateDispersions_BM` *Estimate exon dispersions, using the deprecated "big model" method*

---

## Description

This function estimates for each counting bin of the `ExonCountSet object` a dispersion value. It stores these values in `fData(ecs)$dispersionBeforeSharing`.

## Usage

```
estimateDispersions_BM( object,
         formula=count ~ sample + condition * exon,
         initialGuess=.01, nCores=1, minCount=10,
         maxExon=70, quiet=FALSE, file="")
```

## Arguments

| | |
|---|---|
| `object` | An ExonCountSet object. |
| `formula` | Formula used in the GLM to estimate the dispersion values. The terms in the formula must be design columns of the ExonCountSet object, the l.h.s. must be `count`. |
| `initialGuess` | An initial guess for the dispersion values to initiate the optimization. |
| `nCores` | Number of cores to be used to estimate the dispersions. The `parallel` package must be loaded in order to spread the job onto several cores. |
| `minCount` | Counting bins with less than `minCount` counts (summed over all samples) are skipped in the tests. This reduces computation time, as counting bins with very few counts cannot give a significant signal anyway. For skipped counting bins, the `testable` column in `fData` is set to `FALSE`. |
| `maxExon` | Genes with more than `maxExon` counting bins will be skipped in the test. This option can be useful when otherwise genes with very many counting bins use up extremely long computation time for dispersion estimation and testing for differential exon usage. |
| `quiet` | If `TRUE`, no progress report is shown. In case the session is not an interactive session and a progress report is wanted, include a file name in the parameter `file`. |

`file` A file name to write the progress reports. If `file` is `""`, the output will be written to the standard output connection.

## Details

For the dispersion estimation, we use the Cox-Reid conditional maximum likelihood method of Gordon Smyth et al., which they devised for the `edgeR` package.

## Value

An object of class `ExonCountSet` with dispersion `featureData(object)$dispersion_CR_est)` parameters filled).

## Examples

```
if(suppressWarnings(require("pasilla", quietly=TRUE, character.only=TRUE))){

   data("pasillaExons", package="pasilla")
   pasillaExons <- estimateSizeFactors( pasillaExons )
   pasillaExons <- estimateDispersions( pasillaExons )

}
```

---

# estimateExonDispersionsForModelFrame_BM

*Estimates exon dispersions for the (deprecated) "big model" method*

---

## Description

This function calculates the individual dispersions for each counting bins for a single gene. It is used only for the old, now deprecated, "big model" approach. The function takes as input a model frame generated from the function `modelFrameForGene`.

## Usage

```
estimateExonDispersionsForModelFrame_BM(modelFrame, formula=NULL, mm=NULL,
                                                    muhat=NULL, initialGuess=0.01)
```

## Arguments

`modelFrame` Model frame provided by the function `modelFrameForGene`.

`formula` Formula for the glm used to estimate the dispersions. The factors in the formula must be present in the column names of the model frame. If it is left NULL, the default formula used is "count ~ sample + condition * exon".

`mm` A model matrix for the model frame. If NULL, a model matrix will be created from the parameters "formula" and "modelFrame".

| | |
|---|---|
| muhat | Initial values for the coefficients in the optimization. If NULL, initial values will be calculated using with the dispersion value given by the parameter "initialGuess". |
| initialGuess | An initial guess of the dispersions to initiate the optimization. |

## Value

A vector of exon dispersions.

## See Also

estimateDispersions

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
estimateExonDispersionsForModelFrame_BM(modelFrameForGene(pasillaExons, "FBgn0085442"))
```

---

# estimatelog2FoldChanges

*Fold changes (log2) from the fitted expression values in the GLM.*

---

## Description

This function calculates the fold changes (on log2 scale) between the different conditions. It calculates them from the coefficients of a GLM that fits the read counts to a variable of the experimental design specified by the user (see below, parameter "fitExpToVar").

## Usage

```
estimatelog2FoldChanges(ecs, fitExpToVar="condition",
          denominator="", getOnlyEffects=FALSE, averageOutExpression=TRUE,
          nCores=1, quiet=FALSE, file="")
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| fitExpToVar | A variable contained in design(ecs). The expression values will be fitted to this variable using the the formula "count ~ 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 |
| getOnlyEffects | If TRUE, the raw effects are added as columns to the feature data and any operation (log2) is performed with them. |

averageOutExpression
: The default, TRUE, gives back splicing effects. If FALSE, the gene expression effects won't be substracted.

nCores
: Number of CPU cores to be used to estimate the dispersions. The `multicore` package need to be loaded beforehands to parallelize over several cores.

quiet
: If TRUE, no progress report is shown. In case the session is not an interactive and progress report is wanted, add a file name below.

file
: A file name to write the progress reports. If file="", output will be written to the standard output connection.

## Examples

```
## Not run:
    data("pasillaExons", package="pasilla")
    pasillaExons <- estimateSizeFactors( pasillaExons )
    pasillaExons <- estimateDispersions( pasillaExons )
    pasillaExons <- fitDispersionFunction( pasillaExons )
    pasillaExons <- estimatelog2FoldChanges( pasillaExons )

## End(Not run)
```

---

# estimateSizeFactors *Estimate the size factors for an ExonCountSet*

---

## Description

This function takes the count data from an ExonCountSet object (object), and estimates the size factors as follows: Each column (sample) is divided by the geometric means of the rows. The median of these ratios (skipping the genes with a geometric mean of zero) is used as the size factor for this column.

## Usage

```
## S4 method for signature ExonCountSet
estimateSizeFactors(object)
```

## Arguments

object
: An ExonCountSet object

## Value

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

## Author(s)

Simon Anders, sanders@fs.tum.de

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
        sizeFactors( pasillaExons )
```

---

ExonCountSet *"ExonCountSet", a container for exon count data*

---

## Description

This is the principal class of DEXSeq package.

## Objects from the Class

Objects must be created with the function `newExonCountSet` (q.v.), alternatively the user can call the function `read.HTSeqCounts`, which will call newExonCountset.

## Extends

Class `eSet` (package 'Biobase'), directly. Class `VersionedBiobase` (package 'Biobase'), by class "eSet", distance 2. Class `Versioned` (package 'Biobase'), by class "eSet", distance 3.

## Note

An ExonCountSet object stores the exon counts from high-throughput RNA sequencing experiments. It is the principal object of the DEXSeq package. Some of the slots can be added by the user (see details in newExonCountSet documentation) or alternatively, the user can fill some of the slots by using the HTSeq preprocessing steps and further calling `read.HTSeqCounts`, especially those with the exon annotation data. The other slots will be filled with the analysis.

The ExonCountSet object contains a matrix of non-negative integers which represents sequence counts, with each column representing a sample and and each row a counting bin (i.e., an exon or part of an exon). In the phenoData, the object contains information about the samples, e.g., size factors and design annotations are stored there. The user can also add more information about the other properties of the samples.

An ExonCountSet object can be created just by providing a count matrix, and two vectors of gene and exon identifiers of each of the rows in the matrix. Nevertheless, the visualization plots included in DEXSeq requires additional information about the exons (chromosome, strand, start, end). This information can be added directly after the creation of the ExonCountSet object. If `read.HTSeqCounts` is called to create an ExonCountSet object, this information of the phenoData is inserted directly.

The columns for size factors (in phenoData), dispersion estimates, pvalue and padjust in the featureData are filled later throughout the analysis, when the user calls `estimateSizeFactors`, `estimateDispersions` `fitDispersionFunction`, and `testForDEU`.

## Examples

```
# See the vignette
```

## exonIDs *Accessor for the exonIDs in an ExonCountSet object.*

### Description

This function is an accessor for the exon identifiers for each of the rows in the count table. Note that each exon ID identifies, strictly speaking, not an exon but a counting bin, which may well be just part of an exon. Make sure that the exon IDs are ordered alphanumerically in the gene.

### Usage

```
exonIDs(ecs)
exonIDs(ecs) <- value
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| value | A vector of exon counting bin identifiers, one for each of the rows of the count data. |

### Examples

```
library(DEXSeq)
    data("pasillaExons", package="pasilla")
exonIDs(pasillaExons)
```

## fitDispersionFunction *Fit the mean-variance function.*

### Description

This function fits a parametric model of the mean-dispersion relationship to the per-gene estimates of mean $\hat{\mu}$ and dispersion $\hat{\alpha}$. The parametric model is

$$\alpha(\mu) = \frac{\alpha_1}{\mu} + \alpha_0,$$

where $\mu$ is the mean, $\alpha$ the dispersion and $\alpha_1$ and $\alpha_0$ are two parameters. After this, for each exon, the maximum between the per-gene estimate $\hat{\alpha}$ and the modelled value $\hat{\alpha}_1/\hat{\mu} + \hat{\alpha}_0$ is stored in `fData$dispersion`.

### Usage

```
fitDispersionFunction(ecs)
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |

### Value

An ExonCountSet object with information of the fit included, as well as `fData(ecs)$dispersion` filled.

### Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
        pasillaExons <- estimateDispersions( pasillaExons )
        pasillaExons <- fitDispersionFunction( pasillaExons )
```

---

## geneCountTable *Makes a count table for genes.*

---

### Description

This function returns a count table where each row is a gene and each column is a sample, by adding up the values for each gene's individual counting bins.

### Usage

```
geneCountTable(ecs)
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |

### See Also

DESeq

### Examples

```
data("pasillaExons", package="pasilla")
head(geneCountTable(pasillaExons))
```

## geneIDs *Accessor for the geneIDs in an ExonCountSet object.*

### Description

This function is an accessor for the gene identifiers for each of the rows in the count table.

### Usage

```
geneIDs(ecs)
geneIDs(ecs) <- value
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| value | An factor of gene identifiers, one for each of the rows of the count data. |

### Examples

```
data("pasillaExons", package="pasilla")
head( geneIDs( pasillaExons ) )
```

## makeCompleteDEUAnalysis_BM *Complete differential exon usage analysis*

### Description

This function performs a complete differential exon usage analysis, in the old (now deprecated) "big model" approach, calling all the necessary functions and giving back an ExonCountSet object with p values and p adjusted values.

### Usage

```
makeCompleteDEUAnalysis_BM(ecs,
        formulaDispersion=count ~ sample + condition*exon,
        minCount=10, maxExon=50, formula0=NULL, formula1=NULL,
        FDR=0.1, fitExpToVar="condition", nCores=1, path=NULL,
        color=NULL, color.samples=NULL, quiet=FALSE, file="")
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| formulaDispersion | Formula used in the glm to calculate the dispersion values. The factors on the formula must be present in the design columns of the ExonCountSet object. |
| minCount | Minimum number of counts on an exon for it to be considered in the tests. This significantly increases the speed of the dispersion estimations and testing for differential exon usage. This is supported by the fact that small count exons are less likely of being called significant, so it should not affect the results. |
| maxExon | Genes with more exons than this value will be discarded from the analysis. This is a speed issue. Currently, time of dispersion estimations and testing for differential exon usage increases with number of exons. |
| formula0 | Formula for the NULL model to fit a the glm. The factors must be present in the design columns of the ExonCountSet object. As it is tested for each of the exons, a factor exonID can be added to the formula, so that it will iterate over the exons of the gene fitting the glm for each of them. If it is left in NULL, the default formula is "count~sample+exon+condition" for the NULL model. |
| formula1 | Same as formula0, but for the test model. If it is left in NULL, the default formula will be "count~sample+exon+condition*I(exon==exonID)". If added a factor "exonID", it will iterate over each of the exons of the geneID, e.g. If a geneID contains exons E01, E02, E03,...,EN, and the function is left in the default formula, the function will fit N glms, the last part of the formula will change in the iterations as follows: I(exon==E01), I(exon==E02), I(exon==E03),...,I(exon==EN). |
| FDR | A false discovery rate used to indicate the significant exons. |
| fitExpToVar | A variable contained in the design annotation of the ExonCountSet, the expression values will be fitted to this variable using the formula count~fitExpToVar*exon using a model frame obtained from the function modelFrameForGene. |
| nCores | Number of cores to be used to estimate the dispersions. multicore package must be loaded in order to split the job in several cores. |
| path | A path in the system where to write the report from DEXSeqHTML. If NULL, no report will be created. |
| color | A vector of colors for each of the levels from the factor in the design of the ExonCountSet object indicated by "fitExpToVar". If path is NULL, this parameter will be ignored. |
| 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 level from "fitExpToVar". This option is useful to visualize complex experimental designs. If path is NULL, this parameter will be ignored. |
| quiet | If TRUE, no progress report is shown. In case the session is not an interactive session and progress report is wanted, include a file name in the parameter "file". |
| file | A file name to write the progress reports. If file="", output will be written in the standart output connection. |

## Value

An object of class ExonCountSet.

## Examples

```
data("pasillaExons", package="pasilla")
formuladispersion <- count ~ sample + ( exon + type ) * condition
formula0 <- count ~ sample + type * exon + condition
formula1 <- count ~ sample + type * exon + condition * I(exon == exonID)
pasillaExons <- makeCompleteDEUAnalysis_BM(pasillaExons,
   formulaDispersion=formuladispersion,
   formula0=formula0,
   formula1=formula1)
```

---

# modelFrameForGene *Makes the model frame for a geneID.*

---

## Description

Creates a data frame containing the model frame for a gene with the columns sample, exon, size factors, their respective counts and the design annotation.

## Usage

```
modelFrameForGene(ecs, geneID, onlyTestable=FALSE)
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| geneID | A gene identificator contained in the ExonCountSet object. |
| onlyTestable | If TRUE, only the testable exons will be included in the model frame. Check `fData$testable` for more information. |

## Value

A data frame containing the model frame for a gene.

## Examples

```
data("pasillaExons", package="pasilla")
modelFrameForGene(pasillaExons, "FBgn0085442")
```

`newExonCountSet` *Creates an ExonCountSet object*

## Description

This function creates an ExonCountSet object from a matrix or data.frame of read counts.

## Usage

```
newExonCountSet(countData, design, geneIDs, exonIDs, exonIntervals=NULL,
                          transcripts=NULL)
```

## Arguments

| | |
|---|---|
| `countData` | A matrix or data frame 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. |
| `design` | A factor or data frame with the design annotation (e.g. treatments, or tissue types, or phenotypes, or the like). The length of the factor (or rows in the data frame) has to be equal to the number of columns of the countData matrix, assigning a condition to each sample. All the columns of the design need to be factors. |
| `geneIDs` | A vector of gene identificators ordered according to its respective row in countData. If the gene "x" has four exon counting bins and therefore four rows in countData, then "x" must be four times in the vector. If it is not a factor, it will be converted to one. |
| `exonIDs` | A character vector of exon identifiers ordered according to the rows in countData. The identifiers names can be repeated between genes but not within genes. |
| `exonIntervals` | A data frame with exon annotation information. The number of rows in the data needs to be of the same length as the number of rows in countData. The columns names must contain the values "chr", "start", "end", "strand". This information is only needed for the `plotDEXSeq` function, not for the actual tests. |
| `transcripts` | A character vector of the same length as the rows of the count data containing, for each row in countData, a concatenation of transcript IDs separated by the character ";". This means that if an exon is contained in the transcripts "A", "B" and "C", the field of the row corresponding to that exon should contain "A;B;C". This information is only needed for the `plotDEXSeq` function, not for the actual tests. |

## Value

An object of class ExonCountSet.

## See Also

read.HTSeqCounts

## Examples

```
data("pasillaExons", package="pasilla")
ecs <- newExonCountSet(
   countData=counts(pasillaExons),
   design=design(pasillaExons),
   geneIDs=geneIDs(pasillaExons),
   exonIDs=exonIDs(pasillaExons))
```

---

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(ecs, p = "pvalue", method = perGeneQValueExact)
```

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. fData(ecs) is required to have columns testable and geneID. |
| p | A character string indicating the name of the slot in fData(ecs) 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.

## See Also

See also

## Examples

```
## example code
```

plotDEXSeq *Visualization of the fitted expression, fitted splicing or the normalized counts.*

## Description

The function provides a plot to visualize read count data, the fitted expression, fitted splicing and the results of the test in testForDEU. The fitted values are obtained from fitting the counts values to a certain condition from the design annotation of the glm. See fitExpToVar parameter.

## Usage

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

## Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| geneID | ID of the gene to visualize. |
| FDR | A false discovery rate used to indicate the significant exons. |
| fitExpToVar | A variable contained in the design annotation of the ExonCountSet, the expression values will be fitted to this variable using the formula count ~ fitExpToVar * exon using a model frame obtained from the function modelFrameForGene. |
| 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 function plots the fitted SPLICING estimates from the glm regression. |
| 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("pasillaExons", package="pasilla")
   pasillaExons <- estimateSizeFactors(pasillaExons)
   pasillaExons <- estimateDispersions(pasillaExons)
          pasillaExons <- fitDispersionFunction( pasillaExons )
   plotDEXSeq(pasillaExons, "FBgn0085442")

## End(Not run)
```

---

`plotDispEsts-methods` *Diagnostic plot for dispersion estimates and dispersion fit*

---

## Description

This function generates a diagnostic plot showing the per-exon dispersion estimates and the fit of the dispersion-mean relation.

## Usage

```
## S4 method for signature ExonCountSet
plotDispEsts( object, ymin = NULL, cex = 0.45 )
```

## Arguments

| | |
|---|---|
| `object` | An ExonCountSet object. |
| `ymin` | The lower limit of the y axis. If NULL, a sensible value will be calculated. |
| `cex` | The cex parameter for `plot`. |

## Value

None. A plot is displayed.

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
pasillaExons <- estimateDispersions( pasillaExons )
plotDispEsts( pasillaExons )
```

plotMA-methods *Generate an MA plot*

## Description

This function generates an MA plot.

## Usage

```
## S4 method for signature data.frame
plotMA( object, ylim = NULL,
  colNonSig = "gray32", colSig = "red3", colLine = "#ff000080",
  log = "x", cex=0.45, xlab="mean expression", ylab="log fold change", ... )

## S4 method for signature ExonCountSet
plotMA( object, FDR = 0.1, ... )
```

## Arguments

| | |
|---|---|
| object | either an ExonCountSet or a data.frame. If object is a data.frame, it must contain three columns, the first containing the mean expression values (for the x axis), the second the log fold change (for the y axis) and the third must be a logical vector indicating significance (for the coloring of the dots) |
| FDR | the false discovery rate, i.e., threshold to the adjusted p values, to be used to colour the dots |
| ylim | The limits for the y axis. If missing, an attenpt is made to choose a sensible value. Dots exceeding the limits will be displayed as triangles at the limits, pointing outwards. |
| colNonSig | color to use for non-significant data points |
| colSig | color to use for significant data points |
| colLine | color to use for the horizontal zero line |
| log | which axis should be logarithmic; will be passed to plot |
| cex | The cex parameter for plot. |
| xlab | The x axis label. |
| ylab | The y axis label. |
| ... | Further parameters to be passed through to plot. |

## Examples

```
        ## Not run:
           data("pasillaExons", package="pasilla")
           pasillaExons <- estimateSizeFactors( pasillaExons )
           pasillaExons <- estimateDispersions( pasillaExons )
           pasillaExons <- fitDispersionFunction( pasillaExons )
           pasillaExons <- testForDEU( pasillaExons )
           pasillaExons <- estimatelog2FoldChanges( pasillaExons )
           plotMA( pasillaExons )

    ## End(Not run)
```

---

# prepareAnnotationForDEXSeq-deprecated

*Prepare annotation transcriptDb object for DEXSeq.*

---

## Description

WARNING: This function is deprecated and it has been replaced by the function disjointExons, from the package GenomicFeatures.

## Usage

```
prepareAnnotationForDEXSeq( transcriptDb, aggregateGenes=FALSE, includeTranscripts=TRUE )
```

## Arguments

| | |
|---|---|
| `transcriptDb` | An transcriptDb object. |
| `aggregateGenes` | Logical. Indicates whether two or more genes sharing an exon should be merged into an 'aggregate gene'. If 'no', the exons that can not be assiged to a single gene are ignored. |
| `includeTranscripts` | Logical. Indicates whether the transcript information of each exon should be added. |

## Value

A GRanges object.

## Author(s)

From code kindly provided by Mike Love.

## Examples

```
## Not run:
   library(GenomicFeatures)
  hse <- makeTranscriptDbFromBiomart(biomart="ensembl", dataset="hsapiens_gene_ensembl")
   exonicParts <- prepareAnnotationForDEXSeq( hse )

## End(Not run)
```

---

`read.HTSeqCounts` *Read counts output from HTSeq script.*

---

## Description

This function reads the output files from the HTSeq python scripts dexseq_prepare_annotation.py and dexseq_count.py and gives back an ExonCountSet object.

## Usage

```
read.HTSeqCounts(countfiles, design, flattenedfile=NULL)
```

## Arguments

| | |
|---|---|
| `countfiles` | A string vector containing the output files with the paths from dexseq_count.py. |
| `design` | A vector of factors with information corresponding to each of the countfiles or a data frame design (each column with a factor and each row with its respective sample. If strings are given, they will be converted to factors. |
| `flattenedfile` | An flattened annotation gtf file generated by dexseq_prepare_annotation.py. It is necessary for the visualization of the data but not required to test for alternative exon usage. |

## Value

An ExonCount object.

## Examples

```
library(DEXSeq)
inDir = system.file("extdata", package="pasilla", mustWork=TRUE)
annotationfile = file.path(inDir, "Dmel.BDGP5.25.62.DEXSeq.chr.gff")
samples = data.frame(
   condition = c(rep("treated", 3), rep("untreated", 4)),
   row.names = dir(system.file("extdata", package="pasilla", mustWork=TRUE),
       pattern="fb.txt"),
   stringsAsFactors = TRUE,
   check.names = FALSE
)
```

```
annotationfile = file.path(inDir, "Dmel.BDGP5.25.62.DEXSeq.chr.gff")

   ecs = read.HTSeqCounts(countfiles = file.path(inDir, rownames(samples)),
      design = samples,
      flattenedfile = annotationfile)
```

---

`sizeFactors` *Accessor functions for the sizeFactors information in a ExonCountSet*

---

## Description

The sizeFactors vector assigns to each column of the count data a value, the size factor, such that count values in the columns can be brought to a common scale by dividing by the corresponding size factor.

## Usage

```
## S4 method for signature ExonCountSet
sizeFactors(object)
## S4 replacement method for signature ExonCountSet,numeric
sizeFactors(object) <- value
```

## Arguments

| | |
|---|---|
| `object` | An ExonCountSet |
| `value` | a vector of number, one size factor for each column in the count data |

## Author(s)

Simon Anders, sanders@fs.tum.de

## See Also

`estimateSizeFactors`

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
sizeFactors(pasillaExons)
```

## subsetByGenes *Making an ExonCountSet object from another one with a subset of its genes.*

### Description

Generates a smaller ExonCountSet object containing a subset of genes from another ExonCountSet.

### Usage

```
subsetByGenes(ecs, genes)
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet. |
| genes | Subset of geneIDs used to generate the subset ExonCountSet. |

### Examples

```
data("pasillaExons", package="pasilla")
ecs <- subsetByGenes(pasillaExons, sample(unique(geneIDs(pasillaExons)), 10))
```

## testForDEU *Test for Differential Exon Usage.*

### Description

This function tests for differential exon usage for each of the exons in the object. It stores the results in the fields `fData(ecs)$pvalue` and `fData(ecs)$padjust`.

### Usage

```
testForDEU( ecs,
   formula0 = ~ sample + exon,
   formula1 = ~ sample + exon + condition : exon,
   dispColumn="dispersion", nCores = 1 )
```

### Arguments

| | |
|---|---|
| ecs | An ExonCountSet object. |
| formula0 | Formula for the null model to be used in the GLM fit. |
| formula1 | Formula for the full model to be used in the GLM fit. |
| dispColumn | Column name from the feature data frame from where to take the dispersions. |
| nCores | Number of CPUcores to be used to calculate the $p$-values. The `parallel` package must be loaded to use more than 1 core. |

### Details

The terms in the formulas must be columns of `design(ecs)`.

### Value

An ExonCountSet object with `fData(ecs)$pvalue` and `fData(ecs)$padjust` data slots filled.

### Examples

```
      ## Not run:
         data("pasillaExons", package="pasilla")
         pasillaExons <- estimateSizeFactors( pasillaExons )
         pasillaExons <- estimateDispersions( pasillaExons )
         pasillaExons <- fitDispersionFunction( pasillaExons )
         pasillaExons <- testForDEU( pasillaExons )

## End(Not run)
```

---

## testForDEU_BM *Test for Differential Exon Usage (with the old, deprecated "big model" approach.*

---

### Description

This function tests for differential exon usage for each of the genes in the object. It stores the results in the fields `fData(ecs)$pvalue` and `fData(ecs)$padjust`.

### Usage

```
testForDEU_BM(ecs, formula0=NULL, formula1=NULL, nCores=1, quiet=FALSE, file="")
```

### Arguments

| | |
|---|---|
| `ecs` | An ExonCountSet object. |
| `formula0` | Formula for the null model to be used in the GLM fit. If no formula is given, the default `count ~ sample + exon + condition` is used. See below for details |
| `formula1` | Formula for the full model to be used in the GLM fit. If no formula is given, the default `count ~ sample + exon + condition *I (exon==exonID)` is used. See below for details. |
| `nCores` | Number of CPUcores to be used to calculate the $p$-values. The `parallel` package must be loaded to use more than 1 core. |
| `quiet` | If TRUE, no progress report is shown. In case the session is not an interactive session and progress report is wanted. Change the name of the file. |
| `file` | A file name to write the progress reports. If file="", output will be written in the standard output connection. |

## Details

The terms in the formulas must be columns of `design(ecs)`. In addition, in `formula1`, the variable `exonID` is set to the ID of the currently tested exon counting bin.

See `testGeneForDEU_BM`, which is called for each gene, for further details.

## Value

An ExonCountSet object with `fData(ecs)$pvalue` and `fData(ecs)$padjust` data slots filled.

## See Also

`estimateExonDispersionsForModelFrame`

## Examples

```
      ## Not run:
         data("pasillaExons", package="pasilla")
         pasillaExons <- estimateSizeFactors( pasillaExons )
         pasillaExons <- estimateDispersions_BM( pasillaExons )
         pasillaExons <- fitDispersionFunction( pasillaExons )
         pasillaExons <- testForDEU_BM( pasillaExons )

## End(Not run)
```

---

# testGeneForDEU_BM *Test a single gene for differential exon usage using the old (and deprecated) "big model" approach.*

---

## Description

This function first fits a GLM for the null model, then a GLM for the full model for each exon counting bin. Then, p values are derived with a chi-squared test from the deviance differences between the models.

## Usage

```
testGeneForDEU_BM( ecs, gene, formula0=NULL, formula1=NULL )
```

## Arguments

| | |
|---|---|
| `ecs` | An ExonCountSet object. |
| `gene` | The ID of the gene to be tested for differential exon usage. |
| `formula0` | Formula for the null model. If NULL, the default "`count ~ sample + exon + condition` is used. |
| `formula1` | Formula for the full model. If NULL, the default "`count ~ sample + exon + condition * I(exon==exo` is used. |

## Details

The terms in the formulas must be columns of `design(ecs)`. In addition, in `formula1`, the variable `exonID` is set to the ID of the currently tested exon counting bin, looping through all the counting bins.

The GLMs are of the negative binomial family, using the dispersions from the `dispersion` column in `fData(ecs)`.

## Value

A data frame with columns "deviance", "df" (degrees of freedom) and pvalues from the test.

## See Also

`testForDEU`

## Examples

```
data("pasillaExons", package="pasilla")
pasillaExons <- estimateSizeFactors( pasillaExons )
pasillaExons <- estimateDispersions_BM( pasillaExons )
pasillaExons <- fitDispersionFunction( pasillaExons )
testGeneForDEU_BM(pasillaExons, "FBgn0085442")
```

# Index