Package 'XBSeq'

January 1, 2015

Type Package

Title Test for differential expression for RNA-seq data

Version 1.0

Date 2014-11-19

Author Yuanhang Liu

Maintainer Yuanhang Liu <liuy12@uthscsa.edu>

Description We developed a novel algorithm, XBSeq, where a statistical model was established based on the assumption that observed signals are the convolution of true expression signals and sequencing noises. The mapped reads in non-exonic regions are considered as sequencing noises, which follows a Poisson distribution. Given measureable observed and noise signals from RNA-seq data, true expression signals, assuming governed by the negative binomial distribution, can be delineated and thus the accurate detection of differential expressed genes.

License GPL3

Depends Biobase, pracma, matrixStats, locfit, ggplot2, MASS, methods

R topics documented:

`XBSeq-package` *Differential expression analysis of RNA sequencing data by incorporating non-exonic mapped reads*

Description

We developed a novel algorithm, XBSeq, where a statistical model was established based on the assumption that observed signals are the convolution of true expression signals and sequencing noises. The mapped reads in non-exonic regions are considered as sequencing noises, which follows a Poisson distribution. Given measureable observed and noise signals from RNA-seq data, true expression signals, assuming governed by the negative binomial distribution, can be delineated and thus the accurate detection of differential expressed genes.

Details

Package:	XBSeq
Type:	Package
Version:	1.0
Date:	2014-11-19
License:	>=GPL3
Depends:	Biobase, pracma, matrixStats, locfit, ggplot2

Author(s)

Yuanhang Liu

Maintainer: Yuanhang Liu <liuy12@uthscsa.edu>

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

`adjustScv` *Adjust bias for estimating coefficient of variation (scv)*

Description

The same method from DESeq is adopted to adjust the bias for SCV. We carried out bias correction procedure at estimated signal level rather than observed signal level as in DESeq

Usage

```
adjustScv(scv, nsamples)
```

Arguments

scv	Raw squared coefficient of variation (SCV) for estimated true signal.
nsamples	The number of replicates in each condition

Value

An unbiased estimate for SCV

Author(s)

Yuanhang Liu

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

Examples

```
# This example is adopted from DESeq
   true_mean <- 100
   true_scv <- .1
   nsamples <- 3
   res <- replicate( 2, {
      mySample <- rnbinom( nsamples, mu=true_mean, size=1/true_scv )
      mu_est <- mean( mySample )
      raw_var_est <- var( mySample ) - mean( mySample )
      raw_scv_est <- raw_var_est / mu_est^2
      unbiased_raw_scv_est <- adjustScv( raw_scv_est, 4 )
      c( raw_scv_est = raw_scv_est, unbiased_raw_scv_est = unbiased_raw_scv_est ) } )
   rowMeans( res )
```

---

conditions — *Accessor functions for the 'conditions' information in a XBSeqDataSet object.*

Description

Conditions extract the experimental design information similar as used in DESeq.

Usage

```
## S4 method for signature 'XBSeqDataSet'
conditions(object,...)
## S4 replacement method for signature 'XBSeqDataSet'
conditions(object,...) <- value
```

Arguments

object	a XBSeqDataSet
value	experimental design information
...	Further arguments will be ignored

Value

The experimental design information for a XBSeqDataSet object

Author(s)

Yuanhang Liu

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

Examples

```
cds <- makeExampleXBSeqDataSet()
conditions( cds )
```

---

counts *Access the counts table in XBSeqDataSet object.*

Description

The counts slot holds the count data as a matrix of non-negative integer count values, similar method adopted from DESeq

Usage

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

Arguments

object	a XBSeqDataSet object.
normalized	logical indicating whether or not to divide the counts by the size factors before returning.
value	the integer count matrix for a XBSeqDataSet.

Value

Either the count matrix or the normalized matrix if the argument normalized is set to TRUE

Author(s)

Yuanhang Liu

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

Examples

```
cds <- makeExampleXBSeqDataSet()
head( counts( cds ) )
```

---

dispTable *Access the dispersion information for a XBSeqDataSet object*

Description

A method adopted from DESeq to examine the dispersion information for a XBSeqDataSet object

Usage

```
dispTable(object, ...)
```

Arguments

object	a XBSeqDataSet
...	further argumnts are ignored

Author(s)

Yuanhang Liu

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

See Also

estimateSCV, XBSeqTest

Examples

```
# Example adopted from DESeq
  set.seed(1990)
   conditions <- factor(c('C1','C1','C2','C2'))
   observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
   background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
   Signal <- estimateRealcount(observe_signal, background_noise)
   XB <- newXBSeqDataSet(Signal,conditions)
   XB <- estimateSizeFactorsXBSeq( XB )
   XB <- estimateSCV( XB, observe_signal, background_noise,fitType='local')
   dispTable( XB )
```

`estimateRealcount` *Preliminary step to estimate the underneath true signal based on observed signal and background noise*

Description

Based on the observed signal as well as the background noise, estimate the true signal for each gene.

Usage

```
estimateRealcount(observe, background)
```

Arguments

`observe`	A data.frame which contains the observed signal information; Each row indicates each gene and each column indicates each sample.
`background`	A data.frame which contains the background noise information; The rownames should be the same type of annotation as `observe`, such as gene symbols, refseqs, etc.

Details

The observed signal can be achieved by using HTSeq to count the reads map to exonic regions. The background noise can be extracted by using HTSeq the second time to count the reads map to non-exonic regions, the regions we defined by excluding potential functional elements. The the underneath true signal is estimated by the simple subtraction of observed signal and background noise. The true signal of genes with background noise larger than observed signal will be assigned as 0.

Value

A data.frame contains the estimated true signal for each gene with the same length as observed signal.

Author(s)

Yuanhang Liu

Examples

```
set.seed(1990)
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
Signal <- estimateRealcount(observe_signal,background_noise)
```

estimateSCV *Estimate squared coefficient of variation for each gene*

Description

A similar method is applied to estimate the SCV for each gene based on the method used in DESeq

Usage

```
## S4 method for signature 'XBSeqDataSet'
estimateSCV( object, observe, background,
   method = c( "pooled", "pooled-CR", "per-condition", "blind" ),
   sharingMode = c( "maximum", "fit-only", "gene-est-only" ),
   fitType = c("local","parametric"),
   locfit_extra_args=list(), lp_extra_args=list(),
   modelFrame = NULL, modelFormula = count ~ condition, ... )
```

Arguments

object	a XBSeqDataSet with size factors.
observe	The observed read count from exonic regions
background	The background noise from non-exonic regions
method	There are three ways how the empirical dispersion can be computed:

- pooled - Use the samples from all conditions with replicates to estimate a single pooled empirical dispersion value, called "pooled", and assign it to all samples.
- pooled-CR - Fit models according to modelFormula and estimate the dispersion by maximizing a Cox-Reid adjusted profile likelihood (CR-APL). This method is much slower than method="pooled" but works also with crossed factors (as may occur, e.g., in designs with paired samples). Usually, you will need to specify the model formula, which should be the same as the one used later in the call to nbinomFitGLMs for fitting the full model. Note: The method of using CR-APL maximization for this application has been developed by McCarthy, Chen and Smyth [Nucl. Acid Res., 2012 and been first implemented in edgeR (in 2010). DESeq optimizes the expression for the CR-APL given in McCarthy et al.'s paper, but does not use the weigthed maximum likelihood scheme proposed there.
- per-condition - For each condition with replicates, compute a gene's empirical dispersion value by considering the data from samples for this condition. For samples of unreplicated conditions, the maximum of empirical dispersion values from the other conditions is used. If object has a multivariate design (i.e., if a data frame was passed instead of a factor for the condition argument in newXBSeqDataSet), this method is not available. (Note: This method was called "normal" in previous versions.)
- blind - Ignore the sample labels and compute a gene's empirical dispersion value as if all samples were replicates of a single condition. This can be done even if there are no biological replicates. This method can lead to loss of power; see the vignette for details. The single estimated dispersion condition is called "blind" and used for all samples.

`sharingMode` After the empirical dispersion values have been computed for each gene, a dispersion-mean relationship is fitted for sharing information across genes in order to reduce variability of the dispersion estimates. After that, for each gene, we have two values: the empirical value (derived only from this gene's data), and the fitted value (i.e., the dispersion value typical for genes with an average expression similar to those of this gene). The `sharingMode` argument specifies which of these two values will be written to the `featureData`'s `disp_` columns and hence will be used by the functions `XBSeqTest`

- `fit-only` - use only the fitted value, i.e., the empirical value is used only as input to the fitting, and then ignored. Use this only with very *few* replicates, and when you are not too concerned about false positives from dispersion outliers, i.e. genes with an unusually high variability.
- `maximum` - take the maximum of the two values. This is the conservative or prudent choice, recommended once you have at least three or four replicates and maybe even with only two replicates.
- `gene-est-only` - No fitting or sharing, use only the empirical value. This method is preferable when the number of replicates is large and the empirical dispersion values are sufficiently reliable. If the number of replicates is small, this option may lead to many cases where the dispersion of a gene is accidentally underestimated and a false positive arises in the subsequent testing.

`fitType`

- `parametric` - Fit a dispersion-mean relation of the form `dispersion = asymptDisp + extraPo` via a robust gamma-family GLM. The coefficients `asymptDisp` and `extraPois` are given in the attribute `coefficients` of the `dispFunc` in the `fitInfo` (see below).
- `local` - Use the locfit package to fit a dispersion-mean relation, as described in the DESeq paper.

`locfit_extra_args, lp_extra_args`
(only for `fitType=local`) Options to be passed to the `locfit` and to the `lp` function of the locfit package. Use this to adjust the local fitting. For example, you may pass a value for `nn` different from the default (0.7) if the fit seems too smooth or too rough by setting `lp_extra_agrs=list(nn=0.9)`. As another example, you can set `locfit_extra_args=list(maxk=200)` if you get the error that locfit ran out of nodes. See the documentation of the `locfit` package for details. In most cases, you will not need to provide these parameters, as the defaults seem to work quite well.

`modelFrame` By default, the information in `conditions(object)` or `pData(object)` is used to determine which samples are replicates (see `newXBSeqDataSet`). For `method="pooled"`, a data frame can be passed here, and all rows that are identical in this data frame are considered to indicate replicate samples in `object`. For `method="pooled-CR"`, the data frame is used in the fits. For the other methods, this argument is ignored.

`modelFormula` For `method="pooled-CR"`, this is the formual used for the dispersion fits. For all other methods, this argument is ignored.

`...` extra arguments are ignored

Details

The details regarding which option to choose can be found in the DESeq help page. Generally speaking, if you have less number of replicates (<=3), set `method="pooled"`. Otherwise, try `method="per-condition"`. We revised the code to estimate the variance of the true signal by using variance sum law rather than calculate the variance directly.

Value

The `XBSeqDataSet` cds, with the slots `fitInfo` and `featureData` updated.

Author(s)

Yuanhang Liu

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

Examples

```
set.seed(1990)
 conditions <- factor(c('C1','C1','C2','C2'))
 observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
 background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
 Signal <- estimateRealcount(observe_signal, background_noise)
 XB <- newXBSeqDataSet(Signal,conditions)
 XB <- estimateSizeFactorsXBSeq( XB )
 XB <- estimateSCV( XB, observe_signal, background_noise,fitType='local')
 str( fitInfo( XB ) )
 head( fData( XB ) )
```

---

estimateSizeFactorsForMatrixXBSeq

Internal function to estimate the size factors

Description

The same method is adopted from DESeq to estimate the size factors

Usage

```
estimateSizeFactorsForMatrixXBSeq(counts, locfunc = median)
```

Arguments

`counts`	a matrix of counts information
`locfunc`	a function to compute a location for a sample.

Value

Size factors for each sample in the count matrix.

Author(s)

Yuanhang Liu

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

Examples

```
cds <- makeExampleXBSeqDataSet()
estimateSizeFactorsForMatrixXBSeq( counts(cds) )
```

`estimateSizeFactorsXBSeq` *Estimate size factors for a XBSeqDataSet object*

Description

Same method is adopted from DESeq to estimate the size factors for a XBSeqDataSet

Usage

```
## S4 method for signature 'XBSeqDataSet'
estimateSizeFactorsXBSeq( object, locfunc=median, ... )
```

Arguments

`object`	a XBSeqDataSet
`locfunc`	a function to compute a location for a sample.
`...`	extra arguments are ignored

Value

The `XBSeqDataSet` cds, with the slots `sizefactors` updated.

Author(s)

Yuanhang Liu

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

See Also

`estimateSizeFactorsForMatrixXBSeq`

Examples

```
cds <- makeExampleXBSeqDataSet()
cds <- estimateSizeFactorsXBSeq( cds )
sizeFactors( cds )
```

`fitInfo` *Accessor function for the fitInfo objects in a XBSeqDataSet*

Description

Same method is adopted from DESeq to access the fit information from a XBSeqDataSet

Usage

```
fitInfo( XB, name=NULL )
```

Arguments

`XB`	a XBSeqDataSet
`name`	if `estimateSCV` was called with `method="per-condition"` a name hasd to specified. Try `ls(XB@fitInfo`.

Author(s)

Yuanhang Liu

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

See Also

`estimateSCV`

Examples

```
set.seed(1990)
 conditions <- factor(c('C1','C1','C2','C2'))
 observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
 background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
 Signal <- estimateRealcount(observe_signal, background_noise)
 XB <- newXBSeqDataSet(Signal,conditions)
 XB <- estimateSizeFactorsXBSeq( XB )
 XB <- estimateSCV( XB, observe_signal, background_noise,fitType='local')
 str( fitInfo( XB ) )
```

getCountParams *Extract the mean and variance for a count dataset*

Description

Internal function to extract the mean and variance for each gene based on the normalized counts

Usage

```
getCountParams(counts, sizeFactors)
```

Arguments

counts	one integer count data
sizeFactors	sizeFactors calculated by estimateSizeFactorsXBSeq

Value

a data.frame that contains the mean and variance calculated for each gene

Author(s)

Yuanhang Liu

Examples

```
cds <- makeExampleXBSeqDataSet()
cds <- estimateSizeFactorsXBSeq( cds )
getCountParams(counts(cds),sizeFactors(cds))
```

getSCV *Extract estimation of squared coefficient of variation*

Description

A internal function called by estimateSCV. There is no need to call this function directly

Usage

```
getSCV(means, variances, sizeFactors, fitType = c("parametric", "local"), locfit_extra_args = list
```

Arguments

means	Mean statistics for each gene
variances	Variance statistics for each gene
sizeFactors	sizeFactors for a XBSeqDataSet object
fitType	The method that will be used to fit mean-SCV relation, can either be 'parametric', or 'local'
locfit_extra_args	Futher arguments supplied for "locfit"
lp_extra_args	Futher arguments supplied for lp function in locfit package
adjustForBias	Logical; set whether to carry out bias correction procedure for SCV

Value

a fit object based on means and SCV

Author(s)

Yuanhang Liu

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

See Also

estimateSCV

Examples

```
cds <- makeExampleXBSeqDataSet()
cds <- estimateSizeFactorsXBSeq(cds)
data <- getCountParams( counts(cds), sizeFactors(cds))
SCVf <- getSCV( data$baseMean, data$baseVar, sizeFactors(cds))
```

---

getsignalVars *Estimate variance of the signal based on variance summation law*

Description

Based on variance of observed signal as well as true signal, estimate the variance of the true signal

Usage

```
getsignalVars(counts, bgcounts)
```

Arguments

counts	count data for observed signal
bgcounts	count data for background noise

Value

The estimated variance for true signal

Author(s)

Yuanhang Liu

See Also

estimateSCV

Examples

```
set.seed(1990)
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
data_var <- getsignalVars(observe_signal, background_noise)
```

`makeExampleXBSeqDataSet` *Generate an example XBSeqDataSet object*

Description

The same function is adopted from DESeq. This function returns an example XBSeqDataSet. It is used for the examples in the package help pages.

Usage

```
makeExampleXBSeqDataSet()
```

Value

a XBSeqDataSet that has been constructed as follows: First, true base mean values for 10,000 genes are drawn from an exponential distribution with rate 1/250. Then, certain genes are declared (with probability 0.3 per gene) as truly differentially expressed (tDE). For these genes, the true base mean is split into two values, one for condition "A" and one for condition "B", such that the log2 fold change from "A" to "B" follows a zero-centred normal distribution with standard deviation 2. Then, counts are drawn for each gene for 5 samples, the first three corresponding to condition "A" and the remaining two for condition "B". The counts are drawn from a negative binomial with the specified mean, multiplied by the size factor for the sample, with a constant raw SCV (dispersion) of 0.2 (i.e., a 'size' parameter of 1/0.2). The true size factors are fixed to c(1., 1.3, .7, .9, 1.6).

Author(s)

Yuanhang Liu

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

Examples

```
data <- makeExampleXBSeqDataSet
```

MAplot *Generate maplot after differential expression test*

Description

Generate maplot after differential expression test based on ggplot2

Usage

```
MAplot(stats, ylim, padj = T, pcuff = 0.1, lfccuff = 1, linecol = "red3", xlab = "mean of normalized
```

Arguments

stats	The output of XBSeqTest
ylim	Range of limit for y axis
padj	Whether to use adjusted p value or not
pcuff	Threshold for pvalue
lfccuff	Log fold change cutoff
linecol	Colour of horizontal line
xlab	Lable for x axis
ylab	Lable for y axis

Author(s)

Yuanhang Liu

Examples

```
set.seed(1990)
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
stats <- XBSeq (observe_signal,background_noise,factor(c('C1','C1','C2','C2')) )
MAplot(stats)
```

newXBSeqDataSet *Create a XBSeqDataSet object*

Description

This function creates a XBSeqDataSet object from a matrix or data frame of count data.

Usage

```
newXBSeqDataSet(countData, conditions, sizeFactors = NULL, phenoData = NULL, featureData = NULL)
```

Arguments

countData	A matrix or data frame of count data, each row indicates each gene; each column indicates each sample
conditions	A factor of experimental conditions (or treatments, or tissue types, or phenotypes, or the like). The length of the factor has to be equal to the number of columns of the countData matrix.
sizeFactors	This argument is deprecated. Do not use it.
phenoData	You may pass an AnnotatedDataFrame here to describe the columns of the count matrix. Note that the package always adds two rows (or creates a new AnnotatedDataFrame with only these two rows in case you do not supply one) with names "condition" and "sizeFactor" to store this information.
featureData	You may pass an AnnotatedDataFrame here to describe the rows of the count matrix. The package will just pass through this information without using it. Note that further columns will be added to feature data later, when estimating dispersions.

Details

See also XBSeqDataSet and the documentation of eSet (package Biobase) for the meaning of the other slots, which XBSeqDataSet inherits from eSet (but which the present package does not use).

Value

an object of class XBSeqDataSet

Author(s)

Yuanhang Liu

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

Examples

```
countsTable <- counts( makeExampleXBSeqDataSet() )
cds <- newXBSeqDataSet( countsTable, c( "A", "A", "A", "B", "B" ) )
```

---

newXBSeqDataSetFromHTSeqCount

Creat a XBSeqDataSet object from output of HESeq

Description

The same method is adopted from DESeq. Use this function to start a DESeq analysis if you used htseq-count to count your reads.

Usage

```
newXBSeqDataSetFromHTSeqCount(sampleTable, directory = ".")
```

Arguments

sampleTable	A data frame with three or more columns. Each row describes one sample. The first column is the sample name, the seond column the file name of the count file generated by htseq-count, and the remaining columns are sample meta data. If the meta data consists of only a single column (i.e., three columns in total), this is used as 'condition' factor.
directory	The directory relative to which the filenames are specified.

Value

A XBSeqDataSet object.

Author(s)

Yuanhang Liu

References

See http://www-huber.embl.de/users/anders/HTSeq/ for htseq-count
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

See Also

newXBSeqDataSet

---

plotSCVEsts *Plot estimated SCV*

Description

Plot estimated SCV based on ggplot2

Usage

```
plotSCVEsts(XB, name = NULL, ymin, linecol = "red3", xlab = "mean of normalized counts", ylab = "SC
```

Arguments

XB	A XBSeqDataSet object
name	The name of the fit infromation. Only specify this if you choose method="per-condition"
ymin	The limit of y axis
linecol	The linecolour of the SCV-mean trend
xlab	The lable of x axis
ylab	The lable of y axis

Author(s)

Yuanhang Liu

See Also

estimateSCV

Examples

```
set.seed(1990)
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
Signal <- estimateRealcount(observe_signal, background_noise)
XB <- newXBSeqDataSet(Signal,factor(c('C1','C1','C2','C2')))
XB <- estimateSizeFactorsXBSeq( XB )
XB <-estimateSCV( XB, observe_signal, background_noise)
plotSCVEsts(XB)
```

---

sizeFactors *Access the sizefactor information from a XBSeqDataSet object*

Description

Access the sizefactors assigned to each samples in a XBSeqDataSet object

Usage

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

Arguments

object	a DESeqDataSet object.
value	a numeric vector, one size factor for each column in the count data.

Author(s)

Yuanhang Liu

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

See Also

estimateSizeFactorsXBSeq

Examples

```
cds <- makeExampleXBSeqDataSet()
cds <- estimateSizeFactorsXBSeq( cds )
sizeFactors(cds)
```

XBSeq *Express function to carry out XBSeq analysis*

Description

A wrapper function to carry out XBSeq analysis procedure

Usage

```
XBSeq(observe, background, conditions, method = "pooled", sharingMode = "maximum", fitType = "loca
```

Arguments

observe	A data matrix contains the observed signal
background	A data matrix contains the background noise
conditions	A factor to specify the experimental design
method	Method used to estimate SCV
sharingMode	Mode of sharing of information
fitType	Option to fit mean-SCV relation
pvals_only	Logical; Specify whether to extract pvalues only

Value

id	rownames of XBSeqDataSet
baseMean	The basemean for all genes
baseMeanA	The basemean for condition 'A'
baseMeanB	The basemean for condition 'B'
foldChange	The fold change compare condition 'B' to 'A'
log2FoldChange	The log2 fold change
pval	The p value for all genes
padj	The adjusted p value for all genes

Author(s)

Yuanhang Liu

Examples

```
set.seed(1990)
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
Stats <- XBSeq(observe_signal,background_noise, factor(c('C1','C1','C2','C2') ))
```

XBSeqDataSet-class *Class* "XBSeqDataSet"

Description

Package for use in XBSeq

Objects from the Class

Objects can be created by calls of the form `new("XBSeqDataSet", assayData, phenoData, featureData, experime`

Slots

`fitInfo`: Object of class "environment" ~~

`dispTable`: Object of class "character" ~~

`multivariateConditions`: Object of class "logical" ~~

`assayData`: Object of class "AssayData" ~~

`phenoData`: Object of class "AnnotatedDataFrame" ~~

`featureData`: Object of class "AnnotatedDataFrame" ~~

`experimentData`: Object of class "MIAxE" ~~

`annotation`: Object of class "character" ~~

`protocolData`: Object of class "AnnotatedDataFrame" ~~

`.__classVersion__`: Object of class "Versions" ~~

Extends

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

Methods

conditions `signature(object = "XBSeqDataSet")`: ...

conditions<- `signature(object = "XBSeqDataSet")`: ...

counts `signature(object = "XBSeqDataSet")`: ...

counts<- `signature(object = "XBSeqDataSet", value = "matrix")`: ...

dispTable `signature(object = "XBSeqDataSet")`: ...

dispTable<- `signature(object = "XBSeqDataSet")`: ...

estimateSCV `signature(object = "XBSeqDataSet")`: ...

estimateSizeFactorsXBSeq `signature(object = "XBSeqDataSet")`: ...

sizeFactors `signature(object = "XBSeqDataSet")`: ...

sizeFactors<- `signature(object = "XBSeqDataSet", value = "numeric")`: ...

Author(s)

Yuanhang Liu

XBSeqTest *XBSeq test for differential expression*

Description

The same method is adopted from DESeq for testing differential expression

Usage

```
XBSeqTest(XB, condA, condB, pvals_only = FALSE)
```

Arguments

XB	A XBSeqDataSet object
condA	Factor specified for condition A
condB	Factor specified for condition B
pvals_only	Logical;whether or not only extract p values

Value

id	rownames of XBSeqDataSet
baseMean	The basemean for all genes
baseMeanA	The basemean for condition 'A'
baseMeanB	The basemean for condition 'B'
foldChange	The fold change compare condition 'B' to 'A'
log2FoldChange	The log2 fold change
pval	The p value for all genes
padj	The adjusted p value for all genes

Author(s)

Yuanhang Liu

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

See Also

XBSeq estimateSCV

Examples

```
set.seed(1990)
conditions <- factor(c('C1','C1','C2','C2'))
observe_signal <- matrix(rnbinom(20000,100,0.5),nrow=5000,ncol=4)
background_noise <- matrix(rpois(20000,10),nrow=5000,ncol=4)
Signal <- estimateRealcount(observe_signal, background_noise)
XB <- newXBSeqDataSet(Signal,conditions)
XB <- estimateSizeFactorsXBSeq( XB )
XB <- estimateSCV( XB, observe_signal, background_noise,fitType='local')
Teststas <- XBSeqTest( XB, levels(conditions)[1L], levels(conditions)[2L])
```

Index