DESeq

October 5, 2010

`CountDataSet-class` *Class "CountDataSet" – a container for count data from HTS experiments*

Description

This is the main class for the present package.

Objects from the Class

Objects should be created with calls to `newCountDataSet` (q.v.).

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

Note: This is a summary for reference. For an explanation of the actual usage, see the vignette.

A CountDataSet object stores counts from an HTS data set and offers further slots which are populated during the analysis.

After creation with `newCountDataSet`, a CountDataSet typically contains a count table, i.e., a matrix of integer data, that is accessible with the accessor function `counts`. Each row of the matrix corresponds to a gene (or binding region, or the like), and each colum to an experimental sample. The experimental conditions of the samples are stored in a factor (with one element for each row of the counts matrix), which can be read with the accessor function `conditions`.

In the following analysis steps, further data slots are populated. First, the size factors can be estimated with `estimateSizeFactors`, which are afterwards accessible via `sizeFactors`. Then, the variance functions are estimated with `estimateVarianceFunctions`. The resulting estimates are accessible via the function `rawVarFunc`.

Internally, the mentioned data is stored in slots as follows:

As `CountDataSet` is derived from `eSet`, it has a `phenoData` slots which allows to store sample annotation. This is used to store the factor with the conditions, as a data frame column named `condition`, and to store the size factors, as an numeric data frame column named `sizeFactor`. The user may add further columns to the `phenoData` AnnotatedDataFrame.

The counts table is stored in the `eSet`'s `assayData` locked environment with the name `counts`.

The fits calculated by `estimateVarianceFunctions` are stored as closures in an environment, which is part of the object as slot `rawVarFuncs`. The assignment of the closures' names in the environment to the condition names is stored in a named character vector, which forms the slot `rawVarFuncTable`. For further details on these two slots, see `estimateVarianceFunctions`.

Examples

```
# See the vignette
```

`adjustScvForBias` *Adjust an SCV value for the bias arising when it is calculated from unbiased estimates of mean and variance.*

Description

Assume that a small sample of i.i.d. random variables from a negative binomial distribution is given, and you have obtained unbiased estimates of mean and raw variance. Then, a new bias is introduced when the squared coefficient of variation (SCV) is calculated from these unbiased estimates by dividing the raw variance by the square of the mean. This bias can be calculated by numerical simulation and a pre-calculated adjustment table (or rather a fit through tabulated values) is supplied with the package. The present function uses this to remove the bias from a raw SCV estimate.

This function is used internally in `nbinomTest`. You will rarely need to call it directly.

Usage

```
adjustScvForBias(scv, nsamples)
```

Arguments

`scv`	An estimate for the raw squared coefficient of variation (SCV) for negative binomially distributed data, which has been obtained by dividing an unbiased estimate of the raw variance by the square of an unbiased estimate of the mean.
`nsamples`	The size of the sample used in the estimation.

Value

an unbiased estimate of the raw SCV

Author(s)

Simon Anders

Examples

```
true_mean <- 100
true_scv <- .1
nsamples <- 3
res <- replicate( 1000, {
   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 <- adjustScvForBias( raw_scv_est, 4 )
   c( raw_scv_est = raw_scv_est, unbiased_raw_scv_est = unbiased_raw_scv_est ) } )
rowMeans( res )
```

`conditions` *Accessor function for the conditions information in a CountDataSet*

Description

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

Usage

```
conditions(cds)
```

Arguments

`cds`	a CountDataSet

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
conditions( cds )
```

`counts` *Accessor for the 'counts' slot of a CountDataSet 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

```
counts(cds)
```

Arguments

`cds`	a CountDataSet object

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

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

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

Description

Estimate the size factors for a CountDataSet

Usage

```
estimateSizeFactors(cds)
```

Arguments

`cds`	a CountDataSet

Details

You need to call this function right after `newCountDataSet` unless you have manually specified size factors.

Typically, the function is called with the idiom

```
cds <- estimateSizeFactors( cds )
```

This estimates the size factors and stores the information in the object.

Internally, the function calls `estimateSizeFactorsForMatrix`. See there for more details on the calculation.

Value

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

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

`estimateSizeFactorsForMatrix`

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
sizeFactors( cds )
```

`estimateSizeFactorsForMatrix` *Low-level function to estimate size factors with robust regression.*

Description

Given a matrix or data frame of count data, this function estimates the size factors as follows: Each column 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.

Typically, you will not call this function directly, but use `estimateSizeFactors`.

Usage

```
estimateSizeFactorsForMatrix(counts)
```

Arguments

`counts`	a matrix or data frame of counts, i.e., non-negative integer values

Value

a vector with the estimates size factors, one element per column

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

`estimateSizeFactors`

Examples

```
cds <- makeExampleCountDataSet()
estimateSizeFactorsForMatrix( counts(cds) )
```

`estimateVarianceFunctionForMatrix` *Lower-level function to estimate a raw variance function.*

Description

Usually, you should call `estimateVarianceFunctions` for a CountDataSet instead of calling this function directly. However, if you do not have your data in the form of a CountDataSet object, you may want to call this function directly.

Usage

```
estimateVarianceFunctionForMatrix(counts, sizeFactors,
   locfit_extra_args = list(), lp_extra_args = list())
```

Arguments

counts	a matrix of data frame of count data. All the columns of this matrix will be considered as replicates of the same condition.
sizeFactors	the size factors of the columns, as estimated e.g. with estimateSizeFactorsForMatrix
locfit_extra_args, lp_extra_args	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) or 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. Usually, you will not need to adjuste the fitting parameters, as the defaults seem to work quite fine.

Value

a raw variance function. This is a function that, given a base mean (i.e., the mean of counts divided by size factors), returns a raw variance estimate (which needs to be multiplied with the square of a size factor to get the biological variance on the count scale; to get the full variance, add the shot noise as well).

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
vf <- estimateVarianceFunctionForMatrix( counts(cds), sizeFactors(cds) )
vf( head( counts(cds)[,1] / sizeFactors(cds)[1] ) )
vf( reportSize=TRUE )
```

---

estimateVarianceFunctions

Estimate the variance functions for a CountDataSet.

Description

This function calls, for each condition that has replicates, the lower-level function estimateVarianceFunctionFo to estimate the raw variance function for this condition.

Usage

```
estimateVarianceFunctions(cds, pool = FALSE,
   locfit_extra_args = list(), lp_extra_args = list())
```

Arguments

cds	a CountDataSet with size factors
pool	Ignore the conditions and pool all data to estimate a single raw variance function. Use this option if you have no replicates but be sure to read the caveats in the vignette.
locfit_extra_args, lp_extra_args	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) or 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. Usually, you will not need to adjuste the fitting parameters, as the defaults seem to work quite fine.

Details

Behaviour for pooled=FALSE: The estimated raw variance functions are placed in the environment rawVarFuncs, which is a slot in CountDataSet, using the condition labels as names. A further function, named "_max", is placed there as well, which always return the maximum of all the other functions.

Then, the rawVarFuncTable (q.v.) is filled to assign to each replicated condition the raw variance function estimated for it, and to each condition without replicates, the "_max" function.

Behaviour for pooled=TRUE: A single raw variance function is estimated from all the count data, ignoring the condition labels. It is stored in the rawVarFuncs slot under the name "_pooled". In the rawVarFuncTable, "_pooled" is assigned to all conditions.

Value

The CountDataSet cds, with the slots rawVarFuncs and rawVarFuncTable updated.

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

scvPlot to visualize the result and varianceFitDiagnostics to check the fit

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
vf <- rawVarFunc( cds, "A" )
vf( head( counts(cds)[,1] / sizeFactors(cds)[1] ) )
vf( reportSize=TRUE )
```

getBaseMeansAndVariances

Perform row-wise estimates of base-level means and variances for count data.

Description

This function is called internally by a number of other functions. You will need to call it directly only in very special cases.

Usage

```
getBaseMeansAndVariances(counts, sizeFactors)
```

Arguments

counts	a matrix of data frame of count data. All the columns of this matrix will be considered as replicates of the same condition.
sizeFactors	the size factors of the columns, as estimated e.g. with estimateSizeFactorsForMatrix

Value

A data frame with one row for each row in 'counts' and two columns:

baseMean	The base mean for each row. This is the mean of the counts after they have been divided by the size factors
comp2	The base variance for each row. This is the variance of the counts after they have been divided by the size factors

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
head( getBaseMeansAndVariances( counts(cds), sizeFactors(cds) ) )
```

getVarianceStabilizedData

Perform a variance stabilising transformation (VST) on the count data

Description

This function calculates a variance stabilising transformations (VST) from the raw variance functions and then transforms the count data (after normalization by division by the size factor), yielding a matrix of values which are now approximately homoskedastic. This is useful as input to statistical analyses requiring homoskedasticity.

Usage

```
getVarianceStabilizedData(cds)
```

Arguments

cds	a CountDataSet with estimated variance functions

Details

For each sample (i.e., column of `counts(cds)`), the full variance function is calculated from the raw variance (by scaling according to the size factor and adding the shot noise). The function always uses a pooled estimate of the variance function, i.e., one ignoring conditions. The reciprocal of the square root of the base variance (i.e., the full variance divided by the size factor) is then numerically integrated up, and the integral (approximated by a spline function) evaluated for each count value in the column, yielding a transformed value.

Value

A matrix of the same dimension as the count data, containing the transformed data.

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
vsd <- getVarianceStabilizedData( cds )
colsA <- conditions(cds) == "A"
plot( rank( rowMeans( vsd[,colsA] ) ), genefilter::rowVars( vsd[,colsA] ) )
```

---

makeExampleCountDataSet

make a simple example CountDataSet with random data

Description

This function returns an example CountDataSet. It is used for the examples in the package help pages.

Usage

```
makeExampleCountDataSet()
```

Value

a CountDataSet 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 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).

All these values were chosen to give data that at least somewhat resembles what one might encounter in an actual experiment. Note that this function is not meant to verify the package by simulation. For this purpose the parameters and distribution choices should be more varied.

Author(s)

Simon Anders, anders@embl.de

Examples

```
cds <- makeExampleCountDataSet()
```

---

nbinomTest *Test for differences between the base means for two conditions*

Description

This function tests for differences between the base means of two conditions (i.e., for differential expression in the case of RNA-Seq).

Usage

```
nbinomTest(cds, condA, condB, pvals_only = FALSE)
```

Arguments

cds	a CountDataSet with size factors and raw variance functions
condA	one of the conditions in 'cds'
condB	another one of the conditions in 'cds'
pvals_only	return only a vector of (unadjusted) p values instead of the data frame described below.

Details

See nbinomTestForMatrices for more technical informations

Value

A data frame with the following columns:

`id`	The ID of the observable, taken from the row names of the counts slots.
`baseMean`	The base mean (i.e., mean of the counts divided by the size factors) for the counts for both conditions
`baseMeanA`	The base mean (i.e., mean of the counts divided by the size factors) for the counts for condition A
`baseMeanB`	The base mean for condition B
`foldChange`	The ratio meanB/meanA
`log2FoldChange`	The log2 of the fold change
`pval`	The p value for rejecting the null hypothesis 'meanA==meanB'
`padj`	The adjusted p values (adjusted with 'p.adjust(pval, method="BH")')
`resVarA`	The ratio of the row-wise estimate of the base variance of the counts for condition A, divided by the value predicted with the base variance function from the base mean. If this number is very high, the hit seems to be a variance outlier and might be false.
`resVarB`	The same for condition B.

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
head( nbinomTest( cds, "A", "B" ) )
```

`nbinomTestForMatrices` *Perform row-wise tests for differences between the base means of two count matrices.*

Description

This function is called by `nbinomTest`. Call it directly only if the S4 interface is unsuitable for your task.

Usage

```
nbinomTestForMatrices(countsA, countsB, sizeFactorsA, sizeFactorsB, rawScvA, raw
```

Arguments

countsA	A matrix of counts, where each column is a replicate
countsB	Another matrix of counts, where each column is a replicate
sizeFactorsA	Size factors for the columns of the matrix 'countsA'
sizeFactorsB	Size factors for the columns of the matrix 'countsB'
rawScvA	Raw squared coefficient of variantion (SCV) for 'countsA', a vector with one value per gene
rawScvB	The same for 'countsB'
eps	Precision goal for the p value. This is only a rough guidance with no guarantee of adherence.

Details

See the paper for an exact description of the null hypothesis tested.

Value

A vector of unadjusted p values, one for each row in the counts matrices.

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
colsA <- conditions(cds) == "A"
colsB <- conditions(cds) == "B"
bmvA <- getBaseMeansAndVariances( counts(cds)[,colsA], sizeFactors(cds)[colsA] )
bmvB <- getBaseMeansAndVariances( counts(cds)[,colsB], sizeFactors(cds)[colsB] )
pvals <- nbinomTestForMatrices(
   counts(cds)[,colsA],
   counts(cds)[,colsB],
   sizeFactors(cds)[colsA],
   sizeFactors(cds)[colsB],
   adjustScvForBias(
      rawVarFunc( cds, "A" )( bmvA$baseMean ) / bmvA$baseMean^2,
      length( colsA ) ),
   adjustScvForBias(
      rawVarFunc( cds, "B" )( bmvB$baseMean ) / bmvB$baseMean^2,
      length( colsB ) ) )
names( pvals ) <- row.names( counts(cds) )
head( pvals )
```

newCountDataSet *Create a CountDataSet object*

Description

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

Usage

```
newCountDataSet(countData, conditions, sizeFactors = NULL, phenoData = NULL, fea
```

Arguments

countData	A matrix or data frame of count data, i.e., of non-negative integer values. The rows correspond to observations (e.g., number of reads that were assigned to a gene), the columns correspond to samples (or experiments). Note that biological replicates should each get their own column, while the counts of technical replicates (i.e., several sequencing ruins/lanes from the same sample) have to be summed up into a single column.
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, assigning a condition to each sample. If 'conditions' is not a factor, it will be converted to one.
sizeFactors	The size factors (see sizeFactors). If you have estimated the size factors beforehand, you can pass your estimate here as a numerical vector with as many elements as there are columns in the count data. Usually, however, this field is left blank. Then, you must call function estimateSizeFactors afterwards, which will estimate the size factors and fill in the information.
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.

Details

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

Value

an object of class CountDataSet

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
countsTable <- counts( makeExampleCountDataSet() )
cds <- newCountDataSet( countsTable, c( "A", "A", "A", "B", "B" ) )
```

rawVarFunc *Accessor for raw variance functions*

Description

Given the name of a raw variance function or the name of a condition, get the function from the environment `rawVarFuncs` (which is a slot of CountDataSet).

Usage

```
rawVarFunc(cds, condOrName)
```

Arguments

`cds`	a CountDataSet with variance fucntions (i.e., `estimateVarianceFunctions` has already been called)
`condOrName`	The name of a condition or of a raw variance function. The name is first interpreted as a name in the rawVarFuncs environment. If it is not present there, it is interpreted as the name of a condition and translated into a name of a raw variance function by looking up in in the `rawVarFuncTable`. See `estimateVarianceFunctions` for details.

Value

a raw variance function. This is a function that, given a base mean (i.e., the mean of counts divided by size factors), returns a raw variance estimate. To get the full variance, the raw variance needs to be scales up by multiplying it with the square of the size factor, and the shot noise has to be added.

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

`estimateSizeFactors`

Examples

```
# See example for estimateVarianceFunctions
```

`rawVarFuncTable` *Accessor function for the slot rawVarFuncTable of the class CountDataSet*

Description

Returns the rawVarFuncTable.

Usage

```
rawVarFuncTable(cds)
```

Arguments

`cds`	a CountDataSet object

Value

The rawVarFuncTable is returned. This is a named character vector with one element for each condition, and the condition labels used as names. The value of the element is the name, under which the raw variance function to be used for the corresponding condition can be found in the environment rawVarFuncs (which is a slot of CountDataSet).

The function `estimateVarianceFunctions` fills in this table. The function `rawVarFunc`, and through it several other functions working on CountDataSets, use it to look up which raw variance function to use for which condition.

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

`estimateSizeFactors`

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
rawVarFuncTable( cds )
```

`residualsEcdfPlot` *Produce a diagnostic plot to check the fit of a raw variance function.*

Description

The residuals of the fit should follow a scaled chi-squared distribution. This function calls `varianceFitDiagnostic` to get the cumulative chi-squared probabilities of the residuals and plots their ECDFs, stratified by base means.

Usage

```
residualsEcdfPlot(cds, condition, ncuts = 7)
```

Arguments

`cds`	a CountDataSet with raw variance functions
`condition`	the name of a condition
`ncuts`	the number of base mean strata (i.e, of curves)

Details

As the cumulative chi-square probabilities should be uniform, the ECDF curves should roughly follow the main diagonal (indicated by a green line). It is acceptable if the strata for very low counts deviate from this.

Value

None, but a plot is produced.

See Also

`residualsEcdfPlotFromDiagnostics`, `varianceFitDiagnostics`

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
residualsEcdfPlot( cds, "A" )
```

`residualsEcdfPlotFromDiagnostics` *Produce a diagnostic plot to check the fit of a raw variance function.*

Description

This function produces the same plot as `residualsEcdfPlot`. While `residualsEcdfPlot` takes a CountDataSet as argument and the calls `varianceFitDiagnostics`, the present function expects the output of a call to `varianceFitDiagnostics` (or `varianceFitDiagnosticsForMatrix`) as input.

Usage

```
residualsEcdfPlotFromDiagnostics(fitdiag, ncuts = 7, plotTitle = "Residuals ECDF
```

Arguments

`fitdiag`	a data frame as returned by `varianceFitDiagnostics` or `varianceFitDiagnosticsFor`
`ncuts`	the number of baseMean strata
`plotTitle`	the main title for the plot

Details

See `residualsEcdfPlot`.

Value

None, but a plot is produced.

See Also

`residualsEcdfPlot`, `varianceFitDiagnostics`, `varianceFitDiagnosticsForMatrix`

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
vfd <- varianceFitDiagnostics( cds, "A" )
residualsEcdfPlotFromDiagnostics( vfd )
```

`scvPlot` *Produces a diagnostic plot with the variance estimates, given as squared coefficients of variation (SCV)*

Description

The produced plot shows the estimated variance functions of the given CountDataSet, in the form of the squared coefficient of variation (SCV), i.e., the variance divided by the squared mean. The solid lines are the raw SCV estimates, one per condition. The dashed lines are the full variance estimates for each sample, i.e., the vertical distance between a dashed line and its corresponding solid line (of the same colour) is the shot noise. As the x axis is scaled as base mean (size-adjusted mean), the amount of shot noise depends on the size factor. The solid black line is a density estimate of the base means. Only were a sufficient density of counts is present can a good estimate be expected.

Usage

```
scvPlot(cds, xlim = NULL, ylim = NULL)
```

Arguments

`cds`	a CountDataSet with estimated variance functions
`xlim, ylim`	the plot limits

Value

None; but a plot is produced.

Note

There is still a bug in this function, namely the colours between dashed and solid lines fail to match correctly. Furthermore, the displayed SCV values are not bias corrected.

Author(s)

Simon Anders, sanders@fs.tum.de

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
scvPlot( cds, ylim = c(0, 2) )
```

sizeFactors *Accessor functions for the sizeFactors information in a CountDataSet*

Description

The sizeFactors vector is a factor that 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

```
sizeFactors(cds)
sizeFactors(cds) <- value
```

Arguments

cds	a CountDataSet
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

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
sizeFactors(cds)
```

varianceFitDiagnostics *Generate a data frame with diagnostic information on the fit of a base variance function.*

Description

After fitting base variance function with estimateVarianceFunctions, it is prudent to check the quality of the fit for all or at least a few conditions. This function produces a data frame with useful data for this purpose.

Usage

```
varianceFitDiagnostics(cds, cond)
```

Arguments

cds	a CountDataSet with base variance functions
cond	the name of a condition

Value

A data frame with these columns:

baseMean	The base mean for the condition (see getBaseMeansAndVariances for details).
baseVar	The base variance for the condition as estimated from the counts of the respecive row only (see getBaseMeansAndVariances for details).
fittedRawVar	The raw variance as predicted by the raw variance function when given the baseMean.
fittedBaseVar	The base variance, found by adding the effective base-level shot noise to the raw variance.
pchisq	The cumulative chi-square probability of the residual.

Note

To check the fit, you might want to do one of the following:

Make a doubly-logarithmic scatter plot of baseVar against the baseMean and then add a line with the fittedBaseVar values. The line should follow the points.

- Calculate the squared coefficient of variance (SCV), i.e., baseVar/baseMean^2. Make a scatter plot of the SCV against the (log of the) baseMean and then add a line of the fitted SCV. The line should follow the points. You can also plot 1/baseMean against baseMean as another line to see the shot noise.
- The residuals, i.e., baseVar/fittedBaseVar, should follow a chi-square distribution with k degrees of freedom, after scaling by k (where k is the number of replicates minus 1). (See paper for details). The 'pchisq' column contains the cumulative probabilities (i.e., the result of 'pchisq') of the residuals, and these values should be uniformly distributed. You can check this with a histogram, or more conveniently, with ECDF plot generated by the function residualsEcdfPlot.

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

residualsEcdfPlot

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
vfd <- varianceFitDiagnostics( cds, "A" )
head( vfd )
```

varianceFitDiagnosticsForMatrix *A lower-level function to get fit diagnostics*

Description

Usually, you call varianceFitDiagnostics, which in turn counts varianceFitDiagnosticsForMatrix. If you do not have your data in a CountDataSet object, you may want to call this function directly.

Usage

```
varianceFitDiagnosticsForMatrix(counts, sizeFactors, rawVarFunc)
```

Arguments

counts	a matrix of counts, with columns corresponding to replicates
sizeFactors	the size factors for the columns of the counts matrix
rawVarFunc	the raw variance function estimated from the counts data

Details

See varianceFitDiagnostics for more explanation

Value

The same return value as described for varianceFitDiagnostics.

Author(s)

Simon Anders, sanders@fs.tum.de

See Also

varianceFitDiagnostics,residualsEcdfPlotFromDiagnostics

Examples

```
cds <- makeExampleCountDataSet()
cds <- estimateSizeFactors( cds )
cds <- estimateVarianceFunctions( cds )
colsA <- conditions(cds) == "A"
vfd <- varianceFitDiagnosticsForMatrix(
          counts(cds)[,colsA],
          sizeFactors(cds)[colsA],
          rawVarFunc( cds, "A" ) )
head( vfd )
```

Index