Package 'DSS'

October 9, 2015

Title Dispersion shrinakge for sequencing data.

Version 2.6.0

Date 2014-12-11

Author Hao Wu <hao.wu@emory.edu>

Maintainer Hao Wu <hao.wu@emory.edu>

Depends Biobase,bsseq,splines,methods

Suggests BiocStyle

Description DSS is an R library performing differntial analysis for count-based sequencing data. It detectes differentially expressed genes (DEGs) from RNA-seq, and differentially methylated loci or regions (DML/DMRs) from bisulfite sequencing (BS-seq). The core of DSS is a new dispersion shrinkage method for estimating the dispersion parameter from Gamma-Poisson or Beta-Binomial distributions.

License GPL

biocViews Sequencing, RNASeq, ChIPSeq, DNAMethylation, DifferentialExpression

NeedsCompilation yes

R topics documented:

`DSS-package` *Dispersion shrinkage for sequencing data*

Description

DSS is an R library performing the differential expression analysis for RNA-seq count data. Compared with other similar packages (DESeq, edgeR), DSS implements a new dispersion shrinkage method to estimate the gene-specific biological variance. Extensive simulation results showed that DSS performs favorabily compared to DESeq and edgeR when the variation of biological variances is large.

DSS only works for two group comparison at this time. We plan to extend the functionalities and make it work for more general experimental designs in the near future.

Author(s)

Hao Wu <hao.wu@emory.edu>

`callDML` *Function to detect differntially methylated loci (DML) for two group comparisons of bisulfite sequencing (BS-seq) data.*

Description

This function takes the results from DML testing procedure ('DMLtest' function) and calls DMLs. Regions will CpG sites being statistically significant are deemed as DMLs.

Usage

```
callDML(DMLresult, delta=0.1, p.threshold=1e-5)
```

Arguments

`DMLresult`	A data frame representing the results for DML detection. This should be the result returned from 'DMLtest' function.
`delta`	A threshold for defining DML. In DML testing procedure, hypothesis test that the two groups means are equalis is conducted at each CpG site. Here if 'delta' is specified, the function will compute the posterior probability that the difference of the means are greater than delta,and then call DML based on that.
`p.threshold`	A threshold of p-values for defining significance. Loci with p-values less than this threshold will be deemed DMLs.

Value

A data frame for DMLs. Each row is for a DML. DMLs are sorted by statistical significance. The columns are

`chr`	Chromosome number.
`pos`	Genomic coordinates.
`mu1, mu2`	Mean methylations of two groups.
`diff`	Difference of mean methylations of two groups.
`diff.se`	Standard error of the methylation difference.
`stat`	Wald statistics.
`pval`	P-values. This is obtained from normal distribution.
`fdr`	False discovery rate.
`chr`	Chromosome number.
`pos`	Genomic coordinates.
`meanMethy1, meanMethy2`	Average methylation levels in two conditions.
`diff.Methy`	The difference in the methylation levels between two conditions.

If delta>0, there is an extra column postprob.overThreshold reprsenting the posterior probability of the difference in methylation greater than delta.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

DMLtest, callDMR

Examples

```
## Not run:
require(bsseq)

## first read in methylation data.
path <- file.path(system.file(package="DSS"), "extdata")
dat1.1 <- read.table(file.path(path, "cond1_1.txt"), header=TRUE)
dat1.2 <- read.table(file.path(path, "cond1_2.txt"), header=TRUE)
dat2.1 <- read.table(file.path(path, "cond2_1.txt"), header=TRUE)
dat2.2 <- read.table(file.path(path, "cond2_2.txt"), header=TRUE)

## make BSseq objects
BSobj <- makeBSseqData( list(dat1.1, dat1.2, dat2.1, dat2.2),
  c("C1","C2", "N1", "N2") )

##  DML test
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2"), group2=c("N1","N2"))
```

```
## call DML
dmls <- callDML(dmlTest)
head(dmls)

## call DML with a threshold
dmls2 <- callDML(dmlTest, delta=0.2)
head(dmls2)

## For whole-genome BS-seq data, perform DML test with smoothing
require(bsseqData)
data(BS.cancer.ex)
## takea smallportionof data and test
BSobj <- BS.cancer.ex[10000:15000,]
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2", "C3"), group2=c("N1","N2","N3"),
   smoothing=TRUE, smoothing.span=500)
dmls <- callDML(dmlTest)
head(dmls)



## End(Not run)
```

`callDMR` *Function to detect differntially methylated regions (DMR) for two group comparisons of bisulfite sequencing (BS-seq) data.*

Description

This function takes the results from DML testing procedure ('callDML' function) and calls DMRs. Regions will CpG sites being statistically significant are detected as DMRs. Nearby DMRs are merged into longer ones. Some restrictions including the minimum length, minimum number of CpG sites, etc. are applied.

Usage

```
callDMR(DMLresult, delta=0.1, p.threshold=0.001,
        minlen=50, minCG=3, dis.merge=100, pct.sig=0.5)
```

Arguments

`DMLresult`	A data frame representing the results for DML detection. This should be the result returned from 'DMLtest' function.
`delta`	A threshold for defining DMR. In DML detection procedure, a hypothesis test that the two groups means are equal is conducted at each CpG site. Here if 'delta' is specified, the function will compute the posterior probability that the difference of the means are greater than delta, and then construct DMR based on that.

p.threshold	A threshold of p-values for calling DMR. Loci with p-values less than this threshold will be picked and joint to form the DMRs. See 'details' for more information.
minlen	Minimum length (in basepairs) required for DMR. Default is 50 bps.
minCG	Minimum number of CpG sites required for DMR. Default is 3.
dis.merge	When two DMRs are very close to each other and the distance (in bps) is less than this number, they will be merged into one. Default is 50 bps.
pct.sig	In all DMRs, the percentage of CG sites with significant p-values (less than p.threshold) must be greater than this threshold. Default is 0.5. This is mainly used for correcting the effects of merging of nearby DMRs.

Details

When specifying a 'delta' value, the posterior probability (pp) of each CpG site being DML is computed. Then the p.threshold is applied on 1-pp, e.g., sites with 1-pp<p.threshold is deemed significant. In this case, the criteria for DMR calling is more stringent and users might consider to use a more liberal p.threshold in order to get more regions. This function is rather fast since the computationally intesnsive part is in 'DMLtest'. Users can try different delta and p.threshold to obtain satisfactory results.

Value

A data frame for DMRs. Each row is for a DMR. Rows are sorted by "areaStat", which is the sum of test statistics of all CpG sites in the region. The columns are:

chr	Chromosome number.
start, end	Genomic coordinates.
length	Length of the DMR, in bps.
nCG	Number of CpG sites contained in the DMR.
meanMethy1, meanMethy2	Average methylation levels in two conditions.
diff.Methy	The difference in the methylation levels between two conditions.
areaStat	The sum of the test statistics of all CpG sites within the DMR.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

DMLtest, callDML

Examples

```
## Not run:
require(bsseq)
require(bsseqData)
data(BS.cancer.ex)

## takea smallportionof data and test
BSobj <- BS.cancer.ex[140000:150000,]
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2", "C3"), group2=c("N1","N2","N3"),
   smoothing=TRUE, smoothing.span=500)

## call DMR based on test results
dmrs <- callDMR(dmlTest)
head(dmrs)

## or one can specify a threshold for difference in methylation level
dmrs2 <- callDMR(dmlTest, delta=0.2)
head(dmrs2)

## visualize one DMR
showOneDMR(dmrs[1,], BSobj)


## End(Not run)
```

`dispersion` *Acessor functions for the 'dispersion' slot in a SeqCountData object.*

Description

Dispersion parameter for a gene represents its coefficient of variation of expressions. It characterizes the biological variations.

Usage

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

Arguments

`object`	A SeqCountData object.
`value`	A numeric vector with the same length as number of genes.

Details

If the counts from biological replicates are modeled as negative binomial distribution, the variance (v) and mean (m) should hold following relationship: v=m+m^2*phi, where phi is the dispersion. Another interpretation is that phi represents the biological variations among replicates when underlying expressions are modeled as a Gamma distribution.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

normalizationFactor

Examples

```
data(seqData)
## obtain
seqData=estNormFactors(seqData, "quantile")
seqData=estDispersion(seqData)
dispersion(seqData)

## assign
dispersion(seqData)=rep(0.1, nrow(exprs(seqData)))
```

DMLtest

Function to perform statistical test of differntially methylated loci (DML) for two group comparisons of bisulfite sequencing (BS-seq) data.

Description

This function takes a BSseq object and two group labels, then perform statistical tests for differntial methylation at each CpG site.

Usage

```
DMLtest(BSobj, group1, group2, equal.disp = FALSE, smoothing = FALSE,
  smoothing.method = c("ma", "BSmooth"), smoothing.span = 500, ...)
```

Arguments

BSobj	An object of BSseq class for the BS-seq data.
group1, group2	Vectors of sample names or indexes for the two groups to be tested. See more description in details.

`equal.disp` A flag to indicate whether the dispersion in two groups are deemed equal. Default is FALSE, and the dispersion shrinkages are performed on two conditions independently.

`smoothing` A flag to indicate whether to apply smoothing in estimating mean methylation levels.

`smoothing.method` Method for smoothing. Available options are "ma" for moving average, or "BSmooth" for BSmooth smoothing method. This will be ignored if smoothing=FALSE.

`smoothing.span` The size of smoothing window, in basepairs. Default is 500.

`...` Other parameters for BSmooth function in "bsseq" package.

Details

This is the core function for DML/DMR detection. Tests are performed at each CpG site under the null hypothesis that two groups means are equal. There is an option for applying smoothing or not in estimating mean methylation levels. We recommend to use smoothing=TRUE for whole-genome BS-seq data, and smoothing=FALSE for sparser data such like from RRBS or hydroxyl-methylation data (TAB-seq). If there is not biological replicate, smoothing=TRUE is required. See "Single replicate" section for details.

The BS-seq count data are modeled as Beta-Binomial distribution, where the biological variations are captured by the dispersion parameter. The dispersion parameters are estimated through a shrinakge estimator based on a Bayesian hierarchical model. Then a Wald test is performed at each CpG site.

Due to the differences in coverages, some CpG sites are not covered in both groups, and the test cannot be performed. Those loci will be ignored in test and results will be "NA".

Value

A data frame with each row corresponding to a CpG site. Rows are sorted by chromosome number and genomic coordinates. The columns include:

`chr` Chromosome number.

`pos` Genomic coordinates.

`mu1, mu2` Mean methylations of two groups.

`diff` Difference of mean methylations of two groups.

`diff.se` Standard error of the methylation difference.

`stat` Wald statistics.

`pval` P-values. This is obtained from normal distribution.

`fdr` False discovery rate.

Single replicate

When there is no biological replicate (one sample per treatment group), smoothing=TRUE is required. With the smoothed means, dispersion can still be estimated via the shrinkage estimator. This procedure uses data from neighboring CpG sites as "pseudo-replicate" for estimating biological variance. Results show that the estimation works reasonably well, and the DMR calling results are better than simply using the difference of two smoothed means.

Estimating mean methylation levels

When smoothing=FALSE, the mean methylation levels are estimated based on the ratios of methylated and total read counts, and the spatial correlations among nearby CpG sites are ignored. When smoothing=TRUE, smoothing based on moving average or the BSmooth method is used to estimate the mean methylaion level at each site. Moving average is recommended because it is much faster than BSmooth, and the results are reasonable similar in terms of mean estimation, dispersion estimation, and DMR calling results.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

makeBSseqData, callDML, callDMR

Examples

```
## Not run:
require(bsseq)

## first read in methylation data.
path <- file.path(system.file(package="DSS"), "extdata")
dat1.1 <- read.table(file.path(path, "cond1_1.txt"), header=TRUE)
dat1.2 <- read.table(file.path(path, "cond1_2.txt"), header=TRUE)
dat2.1 <- read.table(file.path(path, "cond2_1.txt"), header=TRUE)
dat2.2 <- read.table(file.path(path, "cond2_2.txt"), header=TRUE)

## make BSseq objects
BSobj <- makeBSseqData( list(dat1.1, dat1.2, dat2.1, dat2.2),
  c("C1","C2", "N1", "N2") )

##  DML test without smoothing
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2"), group2=c("N1", "N2"))
head(dmlTest)

## For whole-genome BS-seq data, perform DML test with smoothing
require(bsseqData)
data(BS.cancer.ex)
## take a small portion of data and test
BSobj <- BS.cancer.ex[10000:15000,]
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2", "C3"), group2=c("N1","N2","N3"),
   smoothing=TRUE, smoothing.span=500)
```

```
head(dmlTest)

## End(Not run)
```

DSS.DE *Perform RNA-seq differential expression analysis in two-group comparison*

Description

This is the top level wrapper function for RNA-seq differential expression analysis in a two-group comparison. Users only need to provide the count matrix and a vector for design, and obtain DE test results.

Usage

```
DSS.DE(counts, design)
```

Arguments

counts	A matrix of integers with rows corresponding to genes and columns for samples.
design	A vector representing the treatment groups. It must be a vector of 0 and 1. The length of the vector must match the number of columns of input count matrix.

Value

A data frame with each row corresponding to a gene. Rows are sorted according to wald test statistics. The columns are:

gene Index	index for input gene orders, integers from 1 to the number of genes.
muA	sample mean (after normalization) for sample A.
muB	sample mean (after normalization) for sample B.
lfc	log fold change of expressions between two groups.
difExpr	differences in expressions between two groups.
stats	Wald test statistics.
pval	p-values.
others	input gene annotations supplied as AnnotatedDataFrame when constructed the SeqCountData object.

Author(s)

Hao Wu <hao.wu@emory.edu>

Examples

```
counts = matrix(rpois(600, 10), ncol=6)
design = c(0,0,0,1,1,1)
result = DSS.DE(counts, design)
head(result)
```

`estDispersion` *Estimate and shrink tag-specific dipsersions*

Description

This function first estimate tag-specific dipsersions using a method of moment estimator. Then the dipsersions are shrunk based a penalized likelihood approach. The function works for general experimental designs.

Usage

```
## S4 method for signature 'SeqCountSet'
estDispersion(seqData, trend=FALSE)
```

Arguments

`seqData`	An object of SeqCountSet class.
`trend`	A binary indicator for modeling the dispersion~expression trend.

Details

The function takes and object of seqCountData class and return the same oject with "dispersion" field filled.

With "trend=TRUE" the dependence of dispersion on mean expressions will be modeled. In that case the shrinkage will be performed conditional on mean expressions.

The function works for multiple factor designs. But option "trend=TRUE" only applicable for single factor experiment.

Author(s)

Hao Wu <hao.wu@emory.edu>

Examples

```
data(seqData)
seqData=estNormFactors(seqData)
seqData=estDispersion(seqData)
head(dispersion(seqData))

## For multiple factor design
data(seqData)
```

```
Y=exprs(seqData)
design=data.frame(gender=c(rep("M",4), rep("F",4)), strain=rep(c("WT", "Mutant"),4))
X=as.data.frame(model.matrix(~gender+strain, data=design))
seqData=newSeqCountSet(Y, X)
seqData=estDispersion(seqData)
head(dispersion(seqData))

## the hypothesis testing for multifactor experiments can be performed
## using edgeR function, with DSS estimated dispersions
## Not run:
library(edgeR)
fit.edgeR <- glmFit(Y, X, lib.size=normalizationFactor(seqData), dispersion=dispersion(seqData))
lrt.edgeR <- glmLRT(fit.edgeR, coef=2)
head(lrt.edgeR$table)

## End(Not run)
```

`estNormFactors` *Estimate normalization factors*

Description

This function estimates normalization factors for the input 'seqCountSet' object and return the same object with normalizationFactor field filled or replaced.

Usage

```
## S4 method for signature 'SeqCountSet'
estNormFactors(seqData, method=c("lr", "quantile", "total", "median"))
```

Arguments

`seqData`	An object of "SeqCountSet" class.
`method`	Methods to be used in computing normalization factors. Currently available options only include methods to compute normalization factor to adjust for sequencing depths. Available options use (1) "lr": using median of logratio of counts. Similar to the TMM method. (2) "quantile" (default): 75th quantile, (3) "total": total counts, or (4) "median": median counts to constuct the normalization factors. From all methods the normalization factor will be a vector with same length as number of columns for input counts.

Value

The same "SeqCountSet" object with normalizationFactor field filled or replaced.

Author(s)

Hao Wu <hao.wu@emory.edu>

Examples

```
data(seqData)
## compare different methods
seqData=estNormFactors(seqData, "lr")
k1=normalizationFactor(seqData)
seqData=estNormFactors(seqData, "quantile")
k2=normalizationFactor(seqData)
seqData=estNormFactors(seqData, "total")
k3=normalizationFactor(seqData)
cor(cbind(k1,k2,k3))

## assign size factor
normalizationFactor(seqData)=k1

## or normalization factor can be a matrix
dd=exprs(seqData)
f=matrix(runif(length(dd), 1,10), nrow=nrow(dd), ncol=ncol(dd))
normalizationFactor(seqData)=f
head(normalizationFactor(seqData))
```

---

makeBSseqData *Create an object of BSseq class from several data frames.*

Description

This is an utility function to merge BS-seq data from replicated experiment and create an object of BSseq class.

After sequence alignment and proper processing, the BS-seq data can be summarized by following information at each C position (mostly CpG sites, with some CH): chromosome number, genomic coordinate, total number of reads covering the position, and number of reads showing methylation at this position. For replicated samples, the data need to be merged based on the chromosome number and genomic coordinates. This function provide such functionality. It takes replicated data as a list of data frames, merged them, and create a BSseq object.

Usage

```
makeBSseqData(dat, sampleNames)
```

Arguments

`dat`	A list of multiple data frames from biological replicates. Each element represents data from one replicate. The data frame MUST contain following columns in correct order: (1) Chromosome number; (2) Genomic coordinates; (3) Read coverage of the position from BS-seq data; (4) Number of reads showing methylation of the position.
`sampleNames`	A vector of characters for the sample names. The length of the vector should match the length of the input list.

Value

An object of 'BSseq' class.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

callDML

Examples

```
require(bsseq)

## first read in methylation data.
path <- file.path(system.file(package="DSS"), "extdata")
dat1.1 <- read.table(file.path(path, "cond1_1.txt"), header=TRUE)
dat1.2 <- read.table(file.path(path, "cond1_2.txt"), header=TRUE)
dat2.1 <- read.table(file.path(path, "cond2_1.txt"), header=TRUE)
dat2.2 <- read.table(file.path(path, "cond2_2.txt"), header=TRUE)

## make BSseq objects
BSobj <- makeBSseqData( list(dat1.1, dat1.2, dat2.1, dat2.2),
  c("C1","C2", "N1", "N2") )

BSobj
sampleNames(BSobj)
```

---

normalizationFactor *Accessor functions for the 'normalizationFactor' slot in a SeqCountData object.*

Description

The normalization factors are used to adjust for technical or biological biases in the sequencing experiments. The factors can either be (1) a vector with length equals to the number of columns of the count data; or (2) a matrix with the same dimension of the count data.

Usage

```
## S4 method for signature 'SeqCountSet'
normalizationFactor(object)
## S4 replacement method for signature 'SeqCountSet,numeric'
normalizationFactor(object) <- value
## S4 replacement method for signature 'SeqCountSet,matrix'
normalizationFactor(object) <- value
```

Arguments

object	A SeqCountData object.
value	A numeric vector or matrix. If it is a vector it must have length equals to the number of columns of the count data. For matrix it must have the same dimension of the count data.

Details

The vector normalization factors are used mostly to correct for sequencing depth from different datasets. The matrix factor applies a different normalizing constant for each gene at each sample to adjust for a broader range of artifacts such as GC content.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

dispersion

Examples

```
data(seqData)
## obtain nomalization factor
seqData=estNormFactors(seqData, "quantile")
normalizationFactor(seqData)

## assign as vector
normalizationFactor(seqData)=rep(1, ncol(exprs(seqData))) ## getan error here

## or assign as a matrix
f=matrix(1, nrow=nrow(exprs(seqData)), ncol=ncol(exprs(seqData)))
normalizationFactor(seqData)=f
```

SeqCountSet-class *Class* "SeqCountSet" *- container for count data from sequencing experiment*

Description

This class is the main container for storing *RNA-seq* data. It is directly inherited fro 'ExpressionSet' class, with two more fields 'normalizationFactor' for normalization factors and 'dispersion' for gene-wise dispersions.

The class for BS-seq data is *BSseq*, which is imported from bsseq package.

Slots

`normalizationFactor`: Normalization factor for counts.

`dispersion`: Gene-wise dispersions.

`experimentData`: See 'ExpressionSet'.

`assayData`: See 'ExpressionSet'.

`phenoData`: See 'ExpressionSet'.

`featureData`: See 'ExpressionSet'.

`annotation`: See 'ExpressionSet'.

`protocolData`: See 'ExpressionSet'.

Extends

Class "ExpressionSet", directly. Class "eSet", by class "ExpressionSet", distance 2. Class "VersionedBiobase", by class "ExpressionSet", distance 3. Class "Versioned", by class "ExpressionSet", distance 4.

Constructor

`newSeqCountSet(counts,designs,normalizationFactor,featureData)`: Creates a 'SeqCountSet' object.

`counts` A matrix of integers with rows corresponding to genes and columns for samples.

`designs` A vector or data frame representing experimental design. The length of the vector or number of rows of the data frame must match the number of columns of input counts. This field can be accessed using 'pData' function.

`normalizationFactor` A vector or matrix of normalization factors for the counts.

`featureData` Additional information for genes as an 'AnnotatedDataFrame' object. This field can be access by using 'featureData' function.

Methods

dispersion, dispersion<- : Access and set gene-wise dispersions.

normalizationFactor, normalizationFactor<- : Access and set normalization factors.

Note

This is similar to 'CountDataSet' in DESeq or 'DGEList' in edgeR.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

dispersion, normalizationFactor

Examples

```
## simulate data from RNA-seq
counts=matrix(rpois(600, 10), ncol=6)
designs=c(0,0,0,1,1,1)
seqData=newSeqCountSet(counts, designs)
seqData
pData(seqData)
head(exprs(seqData))

## multiple factor designs
design=data.frame(gender=c(rep("M",4), rep("F",4)), strain=rep(c("WT", "Mutant"),4))
X=model.matrix(~gender+strain, data=design)
counts=matrix(rpois(800, 10), ncol=8)
seqData=newSeqCountSet(counts, as.data.frame(X))
seqData
pData(seqData)
```

---

seqData *A simulated 'SeqCountData' object.*

Description

The object is created based on simulation for 1000 genes and two treatment groups with 4 replicates in each group.

Usage

```
data(seqData)
```

Examples

```
data(seqData)
seqData
```

---

showOneDMR *Visualze the count data for one DMR*

Description

Given one DMR and an BSseq object, this function generate a multiple panel figure, each for a sample, to visualze the counts. There is a bar at each CpG, the gray bar shows the total coverage, and the black bar shows the methylated count.

Usage

```
showOneDMR(OneDMR, BSobj, ext = 500, ylim)
```

Arguments

OneDMR	A data frame with one row representing one DMR. It must have chr, start, and end columns. This is typically a row from the result generated from callDMR.
BSobj	An object of class BSseq.
ext	The amount (in bps) the plotting region should be extended in both directions.
ylim	Y-axis limit.

Value

This function only generates a figure and has no return values.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

callDMR

Examples

```
## Not run:
require(bsseq)
require(bsseqData)
data(BS.cancer.ex)

## takea small portion of data and test
BSobj <- BS.cancer.ex[140000:150000,]
dmlTest <- DMLtest(BSobj, group1=c("C1", "C2", "C3"), group2=c("N1","N2","N3"),
   smoothing=TRUE, smoothing.span=500)

## call DMR based on test results
dmrs <- callDMR(dmlTest)

## visualize one DMR
showOneDMR(dmrs[1,], BSobj)

## End(Not run)
```

---

waldTest *Perform gene-wise Wald test for two group comparisons for sequencing count data.*

Description

The counts from two groups are modeled as negative binomial random variables with means and dispersions estimated. Wald statistics will be constructed. P-values will be obtained based on Gaussian assumption.

Usage

```
## S4 method for signature 'SeqCountSet'
waldTest(seqData, sampleA, sampleB, equal.var)
```

Arguments

seqData	An object of SeqCountSet class.
sampleA	The sample labels for the first sample to be compared in two-group comparison.
sampleB	The sample labels for the second sample to be compared in two-group comparison.
equal.var	A boolean to indicate whether to use the same or different means in two groups for computing variances in Wald test. Default is FALSE.

Details

The input seqCountData object Must have normalizationFactor and dispersion fields filled, e.g., estNormFactors and estDispersion need to be called prior to this. With group means and shrunk dispersions ready, the variances for difference in group means will be constructed based on Negative Binomial distribution. P-values will be obtained under the assumption that the Wald test statistics are normally distributed. Genes with 0 counts in both groups will be assigned 0 for test statistics and 1 for p-values.

Value

A data frame with each row corresponding to a gene. Rows are sorted according to wald test statistics. The columns are:

gene Index	index for input gene orders, integers from 1 to the number of genes.
muA	sample mean (after normalization) for sample A.
muB	sample mean (after normalization) for sample B.
lfc	log fold change of expressions between two groups.
difExpr	differences in expressions between two groups.
stats	Wald test statistics.
pval	p-values.
others	input gene annotations supplied as AnnotatedDataFrame when constructed the SeqCountData object.

Author(s)

Hao Wu <hao.wu@emory.edu>

Examples

```
data(seqData)
seqData=estNormFactors(seqData)
seqData=estDispersion(seqData)
result=waldTest(seqData, 0, 1)
head(result)
```

Index