Package 'DSS'

April 5, 2014

Title Dispersion shrinakge for sequencing data.

Version 1.8.0

Date 2013-08-06

Imports methods,bsseq,edgeR

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

Depends Biobase, locfdr

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

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 HighThroughputSequencing, RNAseq, ChIPseq, DNAMethylation,DifferentialExpression

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

The replicated BS-seq data are modeled as Beta-Binomial distribution. Similar to that in Gamma-Poisson distributions, the biological variations are captured by the dispersion parameter. This function takes BS-seq data from two conditions, and estimate the dispersion parameters through a Bayesian hierarchical model. Then a Wald test is constructed to test the differntial methylation.

Usage

```
callDML(BS1, BS2, equal.disp = FALSE, threshold = 0)
```

Arguments

BS1	An object of BSseq class for the BS-seq data from the first condition.
BS2	Anobject of BSseqclass for the BS-seq data from the second condition.
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.
threshold	A threshold for defining DML. See details for more description.

Details

By default, a hypothesis testing that the two groups means are equal is conducted at each CpG site. The spatial correlations among the CpG sites are ignored in this function, and the tests are conducted independently at each CpG site. If 'threshold' is specified, the function will also report the posterior probability that the difference of the means are greater than the threshold.

Due to the differences in coverages, some CpG sites are not covered at all replicates. For a CpG sites, if it's only covered by one replicate, the test cannot be performed because the within group variances cannot be estimated. For those loci the 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.

Author(s)

Hao Wu <hao.wu@emory.edu>

See Also

makeBSseqData

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
BS1 <- makeBSseqData( list(dat1.1, dat1.2), paste("cond1",1:2,sep=".") )
BS2 <- makeBSseqData( list(dat2.1, dat2.2), paste("cond2",1:2,sep=".") )

## call DML
```

```
result <- callDML(BS1, BS2)
head(result)

## 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)))
```

`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("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) "quantile" (default): 75th quantile, (2) "total": total counts, or (3) "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, "quantile")
k1=normalizationFactor(seqData)
seqData=estNormFactors(seqData, "total")
k2=normalizationFactor(seqData)
cor(k1,k2)

## 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 a little 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 should contain following files: (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)

## make BSseq objects
BS1 <- makeBSseqData( list(dat1.1, dat1.2), paste("cond1",1:2,sep=".") )
BS1
```

`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 data from sequencing technology. It is directly inherited fro 'ExpressionSet' class, with two more fields 'normalizationFactor' for normalization factors and 'dispersion' for gene-wise dispersions.

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

```
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
```

`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