edgeR

October 5, 2010

`DGEList-class` *Digital Gene Expression data - class*

Description

A simple list-based class for storing read counts from digital gene expression technologies and other important information for the analysis of DGE data.

Slots/List Components

Objects of this class contain (at least) the following list components:

`counts`: numeric matrix containing the read counts.

`samples`: data.frame containing the library size and group labels.

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class. `DGEList` objects also have a `show` method.

Author(s)

Mark Robinson

See Also

`DGEList`

DGEList *DGEList Constructor*

Description

A function to create a DGEList object from a table of counts (rows=features, columns=samples), group indicator for each column, library size (optional) and a table of annotation (optional)

Usage

```
DGEList(counts = matrix(0, 0, 0), lib.size = NULL, group = factor(), genes = NUL
```

Arguments

counts	numeric matrix containing the read counts.
lib.size	numeric vector containing the total to normalize against for each sample (optional)
group	vector giving the experimental group/condition for each sample/library
genes	data frame containing annotation information for the tags/transcripts/genes for which we have count data (optional).
remove.zeros	whether to remove rows that have 0 total count; default is FALSE so as to retain all information in the dataset

Details

If no lib.size argument is passed to the constructor, the column totals are used.

The optional genes argument is meant to be an annotation data.frame, with rows matching those in the counts argument.

Value

a DGEList object

Author(s)

Mark Robinson

See Also

DGEList

Examples

```
y <- matrix(rnbinom(10000,mu=5,size=2),ncol=4)
d <- DGEList(counts=y, group=rep(1:2,each=2), lib.size=colSums(y))
```

EBList-class *Differential Expression of Digital Gene Expression data - class*

Description

A simple list-based class for storing results of the approximate empirical Bayes rule parameters

Slots/List Components

Objects of this class contain the following list components: `sigma2.0.est`: numeric scale `sigma_0^2` estimate. `alpha`: numeric scalar alpha estimate. `scores`: numeric scalar (likelihood) score. `infos`: numeric vector containing the (likelihood) information for each tag. `quantileAdjusted`: list from output of `quantileAdjust`).

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class. `EBList` objects also have a `show` method.

Author(s)

Mark Robinson, Davis McCarthy

Tu102 *Raw Data for Several SAGE Libraries from the Zhang 1997 Science Paper.*

Description

SAGE dataset for 2 tumour samples, 2 normal samples.

Usage

```
data(Tu102)
```

Format

Data frames with 22713, 18794, 16270 and 17703 observations (for Tu102, Tu98, NC2, NC1, respectively) on the following 2 variables.

`Tag_Sequence` a character vector

`Count` a numeric vector

Source

Zhang et al. (1997) Gene Expression Profiles in Normal and Cancer Cells. *Science*, 276, 1268-72.

`alpha.approxeb` *Estimate the Prior Weight, Alpha*

Description

Estimate the prior weight, using an approximate empirical Bayes rule

Usage

```
alpha.approxeb(object, verbose=TRUE)
```

Arguments

`object`	`DGEList` object containing the raw counts with elements `counts` (table of counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
`verbose`	whether to write comments, default `true`

Details

An older function, no longer called by the functions recommended to carry out analysis of DGE data, namely `estimateCommonDisp`, `estimateTagwiseDisp` and `exactTest`. Estimation of the prior weight should now be done using `estimateSmoothing`, if at all.

Value

`EBList` object with elements `sigma2.0.est` (numeric scale `sigma_0^2` estimate), `alpha` (estimate for the prior weight, alpha), `scores` (likelihood scores), `infos` (Fisher expected information), `quantileAdjusted` (list from output of `quantileAdjust`)

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
alpha<-alpha.approxeb(d)
```

`approx.expected.info` *Approximate Expected Information (Fisher Information)*

Description

Using a linear fit (for simplicity), the expected information from the conditional log likelihood of the dispersion parameter of the negative binomial is calculated over all genes.

Usage

```
approx.expected.info(object, d, pseudo, robust = FALSE)
```

Arguments

`object`	`DGEList` object containing the raw counts with (at least) elements `counts` (table of counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
`d`	numeric vector giving the delta parameter for negative binomial - `phi/(phi+1)` ; either of length 1 or of length equal to the number of tags/transcripts (i.e. number of rows of `object$counts`.
`pseudo`	numeric matrix of pseudocounts from output of `estimateDispIter`
`robust`	logical on whether to use a robust fit, default `FALSE`

Value

numeric vector of approximate values of the Fisher information for each tag/transcript (with length same as the number of rows of the original counts)

Author(s)

Mark Robinson

See Also

This function is used in the algorithm for estimating an appropriate amount of smoothing for the dipsersion estimates carried out by `estimateSmoothing`.

Examples

```
set.seed(0)
y<-matrix(rnbinom(40,size=1,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
qA<-estimateDispIter(d,prior.n=10)
exp.inf<-approx.expected.info(d,1/(1 + qA$dispersion[1]),qA$pseudo)
```

`calcNormFactors` *Calculates Normalization Factors for a Matrix of Count Data*

Description

Using a reference sample, calculate the normalization factors, over and above accounting for library size.

Usage

```
calcNormFactors(dataMatrix, refColumn = 1, logratioTrim = .3, sumTrim = 0.05, do
```

Arguments

`dataMatrix`	matrix of raw (read) counts
`refColumn`	column to use as reference
`logratioTrim`	amount of trim to use on log-ratios ("M" values)
`sumTrim`	amount of trim to use on the combined absolute levels ("A" values)
`doWeighting`	logical, whether to compute (asymptotic binomial precision) weights
`Acutoff`	cutoff on "A" values to use before trimming

Details

The weighted trimmed mean of M values (to the reference) is used as the normalization factor, where the weights are from the delta method on Binomial data (more details to come).

The normalization factor for the reference sample will always be 1.

Value

vector with length `ncol(dataMatrix)` giving the relative normalization factors

Author(s)

Mark Robinson

Examples

```
d <- matrix( rpois(1000, lambda=5), nrow=200 )
f <- calcNormFactors(d)
```

commonCondLogLikDerDelta
Conditional Log-Likelihoods in Terms of Delta

Description

Common conditional log-likelihood parameterized in terms of delta (`phi / (phi+1)`)

Usage

```
commonCondLogLikDerDelta(y, delta, der = 0, doSum = FALSE)
```

Arguments

`y`	list with elements comprising the matrices of count data (or pseudocounts) for the different groups
`delta`	delta (`phi / (phi+1)`) parameter of negative binomial
`der`	derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)
`doSum`	logical, whether to sum over samples or not (default `FALSE`

Details

The common conditional log-likelihood is constructed by summing over all of the individual tag conditional log-likelihoods. The common conditional log-likelihood is taken as a function of the dispersion parameter (`phi`), and here parameterized in terms of delta (`phi / (phi+1)`). The value of delta that maximizes the common conditional log-likelihood is converted back to the `phi` scale, and this value is the estimate of the common dispersion parameter used by all tags.

Value

numeric scalar of function/derivative evaluated at given delta

Author(s)

Davis McCarthy

See Also

`estimateCommonDisp` is the user-level function for estimating the common dispersion parameter.

Examples

```
counts<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=counts,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
y<-splitIntoGroups(d)
ll1<-commonCondLogLikDerDelta(y,delta=0.5,der=0,doSum=FALSE)
ll2<-commonCondLogLikDerDelta(y,delta=0.5,der=1)
```

`condLogLikDerDelta` *Conditional Log-Likelihood in Terms of Delta*

Description

Conditional negative binomial log-likelihood parameterized in terms of delta (`phi / (phi+1)`)

Usage

```
condLogLikDerDelta(y, delta, grid = TRUE, der = 1, doSum = TRUE)
```

Arguments

`y`	matrix with count data (or pseudocounts)
`delta`	delta (`phi / (phi+1)`)parameter of negative binomial
`grid`	logical, whether to calculate a grid over the values of delta
`der`	derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)
`doSum`	logical, whether to sum over samples or not (default `TRUE`

Details

This function computes the individual tag conditional log-likelihood for each tag. It is necessary for computing both the common conditional log-likelihood and the weighted conditional log-likelihood, which are used to find the common and tagwise (moderated) estimates of the dipsersion parameter. The delta scale for convenience (delta is bounded between 0 and 1).

Value

vector or matrix of function/derivative evaluations

Author(s)

Mark Robinson, Davis McCarthy

See Also

`commonCondLogLikDerDelta` and `weightedCondLogLikDerDelta` rely on `condLogLikDerDelta`, and at a user level, `estimateCommonDisp` and `estimateTagwiseDisp` are used to estimate the common and (moderated) tagwise dispersion estimates, respectively. `condLogLikDerDelta` calls `condLogLikDerSize`, the function that does the mathematical calculations.

Examples

```
y1<-matrix(rnbinom(10,size=1,mu=10),nrow=5)
v1<-seq(.1,.9,length=9)
ll1<-condLogLikDerDelta(y1,v1,grid=TRUE,der=0,doSum=FALSE)
ll2<-condLogLikDerDelta(y1,delta=.5,grid=FALSE,der=0)
```

`condLogLikDerSize` *Log-Likelihood of the Common Dispersion for a Single Equalized Group*

Description

Derivatives of the conditional negative-binomial log-likelihood (for each tag/transcript) with respect to the common dispersion parameter, for a single group of replicate libraries of the same size. Parameterized in terms of size or precision (`1/phi`).

Usage

```
condLogLikDerSize(y, r, der=1)
```

Arguments

`y`	matrix of (pseudo) count data
`r`	size parameter of negative binomial distribution
`der`	order of derivative required, either 0 (the function), 1 (first derivative) or 2 (second derivative)

Details

The library sizes must be equalized before running this function. This function carries out the actual mathematical computations for the conditional log-likelihood and its derivatives, calculating the conditional log-likelihood for each tag/transcript.

Value

vector of function/derivative evaluations, one for each transcript

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y <- matrix(rnbinom(10,size=1,mu=10),nrow=5)
condLogLikDerSize(y,r=1,der=1)
```

de4DGE *Compute Moderated Differential Expression Scores for Digital Gene Expression (DGE) Data*

Description

Runs weighted likelihood calculation for moderated estimates of dispersion, and tests for differences in 'tag' abundance between groups

Usage

```
de4DGE(object,prior.n=10, disp.init=NULL, doPoisson=FALSE, useCommonDisp=TRUE, v
```

Arguments

object	DGEList object containing (at least) elements counts (matrix: rows-tags/genes, columns-libraries), lib.size, group indicating class
prior.n	numeric scalar for the smoothing parameter that indicates the weight to put on the common likelihood compared to the individual tag's likelihood; default 10 means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion
disp.init	initialized value(s) of the dispersion parameter, can be a common value or tag/genewise values; if NULL, then the common value on Poisson-adjusted counts is used
doPoisson	logical, if TRUE then use a Poisson model rather than Negative Binomial to analyse the data; default FALSE
useCommonDisp	logical, if TRUE then the common dispersion estimate is used for all tags/genes, otherwise tag/genewise dispersion parameters are estimated; default TRUE
verbose	logical, whether to write comments, default TRUE

Details

An older function, no longer included in the recommended analysis pathway for DGE data. Instead, see estimateCommonDisp, estimateTagwiseDisp and exactTest.

Value

deDGEList object with elements

conc	list containing concentration estimates
dispersion	estimates of dispersion parameter)
pseudo	numeric matrix of pseudocounts generated by quantileAdjust
group	vector or factor indicating the experimental class of each sample
M	numeric scalar giving the library size to which counts are adjusted; the geometric mean of the original library sizes

Author(s)

Mark Robinson, Davis McCarthy

References

Robinson MD, Smyth GK. 'Small-sample estimation of negative binomial dispersion, with applications to SAGE data.' Biostatistics. 2008 Apr;9(2):321-32.

Robinson MD, Smyth GK. 'Moderated statistical tests for assessing differences in tag abundance.' Bioinformatics. 2007 Nov 1;23(21):2881-7.

Examples

```
# generate raw counts from NB, create list object
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))

# estimate common dispersion, find smoothing parameter and call main procedure to find di
d<-estimateCommonDisp(d)
prior.n<-estimateSmoothing(d)
ms<-de4DGE(d,prior.n=prior.n,disp.init=d$common.dispersion)
```

---

de4DGEList-class *differential expression for Digital Gene Expression data - class*

Description

A simple list-based class for storing results of differential expression analysis for DGE data

Slots/List Components

Objects of this class contain the following list components: `ps`: list containing estimates of `p` parameter, the expression proportions for the tags/genes. `dispersion`: numeric vector of dispersion parameter (`phi`, the negative binomial dispersion). `pseudo`: numeric matrix with the pseudo-counts. `group`: vector giving the experimental group/condition. `M`: numeric scalar with the library size that pseudo counts are mapped to.

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class. `deDGEList` objects also have a `show` method.

Author(s)

Mark Robinson, Davis McCarthy

deDGE *Compute Moderated Differential Expression Scores for Digital Gene Expression (DGE) Data*

Description

Runs weighted likelihood calculation for moderated estimates of dispersion, and tests for differences in 'tag' abundance between groups

Usage

```
deDGE(object,alpha=500,doPoisson=FALSE,verbose=TRUE)
```

Arguments

object	DGEList containing elements counts (matrix: rows-tags, columns-libraries), lib.size, group indicating class
alpha	weight to put on the individual tag's likelihood
doPoisson	logical, whether to fit Poisson model instead of Negative Binomial, default FALSE
verbose	logical, whether to write comments, default TRUE

Details

An older function, no longer included in the recommended analysis pathway for DGE data. Instead, see estimateCommonDisp, estimateTagwiseDisp and exactTest.

Value

deDGEList with elements ps (list containing proportion estimates), r (estimates of 1/overdispersion), pseudo (pseudocounts generated by quantileAdjust), group (indicating class of each sample), M (geometric mean of library sizes)

Author(s)

Mark Robinson, Davis McCarthy

References

Robinson MD, Smyth GK. 'Small-sample estimation of negative binomial dispersion, with applications to SAGE data.' Biostatistics. 2008 Apr;9(2):321-32.

Robinson MD, Smyth GK. 'Moderated statistical tests for assessing differences in tag abundance.' Bioinformatics. 2007 Nov 1;23(21):2881-7.

Examples

```
# generate raw counts from NB, create list object
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))

# find alpha and call main procedure to find differences
alpha<-alpha.approxeb(d)
ms<-deDGE(d,alpha=alpha$alpha)
```

---

deDGEList-class *differential expression of Digital Gene Expression data - class*

Description

A simple list-based class for storing results of differential expression analysis for DGE data

Slots/List Components

Objects of this class contain the following list components:

`table`: data frame containing the log-concentration (i.e. expression level), the log-fold change in expression between the two groups/conditions and the exact p-value for differential expression, for each tag.

`comparison`: vector giving the two experimental groups/conditions being compared.

`genes`: a data frame containing information about each transcript (can be `NULL`).

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class. `deDGEList` objects also have a `show` method.

Author(s)

Mark Robinson, Davis McCarthy

---

edgeR-package *Empirical analysis of digital gene expression data in R*

Description

edgeR is a library for the analysis of digital gene expression data arising from RNA sequencing technologies such as SAGE, CAGE, Tag-seq or RNA-seq, with emphasis on testing for differential expression.

Particular strengths of the package include the ability to estimate biological variation between replicate libraries, and to conduct exact tests of significance which are suitable for small counts. The package is able to make use of even minimal numbers of replicates.

A User's Guide is available as well as the usual help page documentation for each of the individual functions.

The library implements statistical methodology developed by Robinson and Smyth (2007, 2008).

Author(s)

Mark Robinson <mrobinson@wehi.edu.au>, Davis McCarthy <dmccarthy@wehi.edu.au>, Gordon Smyth

References

Robinson MD and Smyth GK (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* 23, 2881-2887

Robinson MD and Smyth GK (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9, 321-332

Robinson MD, McCarthy DJ and Smyth GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26, 139-140

---

equalizeLibSizes *Quantile Adjustment to Equalize Library Sizes for a Fixed Value of the Dispersion Parameter*

Description

A function that uses a NB quantile-to-quantile method to adjust the libraries of counts so that library sizes are equal for a fixed value of the dispersion parameter.

Usage

```
equalizeLibSizes(object, disp=0, N=exp(mean(log(object$samples$lib.size))),null.
```

Arguments

`object`	`DGEList` object containing the raw counts with elements `counts` (table of counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
`disp`	numeric scalar or vector of `dispersion` parameters; if a scalar, then a common dispersion parameter is used for all tags
`N`	numeric scalar, the library size to normalize to; default is the geometric mean of the original library sizes
`null.hypothesis`	logical, whether to calculate the `input.mean` and `output.mean` under the null hypothesis; default is `FALSE`

Details

The function `equalizeLibSizes` provides the necessary framework and calculations to call `q2qnbinom`, for given value(s) of the dispersion parameter. The function `q2qnbinom` actually generates the pseudocounts, the counts that have been adjusted for normalized library sizes. These pseudocounts are required to estimate the dispersion parameter, as the methods used by `estimateCommonDisp` and `estimateTagwiseDisp` rely on the assumption of equal library sizes. This function calls `estimatePs` to estimate the expression proportion for each tag, which is needed to calculate the `input.mean` and `output.mean` for each tag, which are passed to `q2qnbinom` along with the unadjusted counts and the fixed value(s) for the dispersion parameter.

Value

A list with elements

pseudo	numeric matrix of pseudocounts, i.e. adjusted counts for equalized libraries
conc	list with elements conc.common (vector giving overall proportion/concentration for each tag), and conc.group (matrix with columns giving estimates of tag/gene concentrations (proportion of total RNA for that group that that particular tag/gene contributes) for different groups); output from estimatePs
N	normalized library size

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y<-matrix(rnbinom(10000,size=2,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000,1010),2))
ps<-estimatePs(d,r=2)
q2q.out<-equalizeLibSizes(d,disp=0.5,null.hypothesis=FALSE)
```

---

estimateCommonDisp *Estimates the Negative Binomial Common Dispersion by Maximizing the Negative Binomial Conditional Common Likelihood*

Description

Maximizes the negative binomial conditional common likelihood to give the estimate of the common dispersion across all tags for the unadjusted counts provided.

Usage

```
estimateCommonDisp(object, tol=1e-06, rowsum.filter=5)
```

Arguments

object	DGEList object with (at least) elements counts (table of unadjusted counts), and samples (vector indicating group) and lib.size (vector of library sizes)
tol	numeric scalar providing the tolerance to be passed to optimize; default value is 1e-06
rowsum.filter	numeric scalar giving a value for the filtering out of low abundance tags in the estimation of the common dispersion. Only tags with total sum of counts above this value are used in the estimation of the common dispersion. Low abundance tags can adversely affect the estimation of the common dispersion, so this argument allows the user to select an appropriate filter threshold for the tag abundance.

Details

The method of conditional maximum likelihood assumes that library sizes are equal, which is not true in general, so pseudocounts (counts adjusted so that the library sizes are equal) need to be calculated. The function `equalizeLibSizes` is called to adjust the counts using a quantile-to-quantile method, but this requires a fixed value for the common dispersion parameter. To obtain a good estimate for the common dispersion, pseudocounts are calculated under the Poisson model (dispersion is zero) and these pseudocounts are used to give an estimate of the common dispersion. This estimate of the common dispersion is then used to recalculate the pseudocounts, which are used to provide a final estimate of the common dispersion.

Value

`estimateCommonDisp` produces an object of class `DGEList` with the following components.

`common.dispersion`
: estimate of the common dispersion; the value for `phi`, the dispersion parameter in the NB model, that maximizes the negative binomial common likelihood on the `phi` scale

`counts`
: table of unadjusted counts

`group`
: vector indicating the group to which each library belongs

`lib.size`
: vector containing the unadjusted size of each library

`pseudo.alt`
: table of adjusted counts; quantile-to-quantile method (see `q2qnbinom`) used to adjust the raw counts so that library sizes are equal; adjustment here done under the alternative hypothesis that there is a true difference between groups

`conc`
: list containing the estimates of the concentration of each tag in the underlying sample; `conc$p.common` gives estimates under the null hypothesis of no difference between groups; `conc$p.group` gives the estimate of the concentration for each tag within each group; concentration is a measure of abundance and thus expression level for the tags

`common.lib.size`
: the common library size to which the count libraries have been adjusted

Author(s)

Mark Robinson, Davis McCarthy

References

Robinson MD and Smyth GK (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9, 321-332

See Also

`estimateTagwiseDisp` can be used to estimate a value for the dispersion parameter for each tag/transcript. The estimates are stabilized by squeezing the estimates towards the common value calculated by `estimateCommonDisp`.

Examples

```
y<-matrix(rnbinom(1000,mu=10,size=2),ncol=4)
d<-DGEList(counts=y,group=c(1,1,2,2),lib.size=c(1000:1003))
cmdisp<-estimateCommonDisp(d)
```

`estimateDispIter` *Normalizes a Dataset Using Quantile Adjustment and Iteratively Estimates the Dispersion Parameter*

Description

The function equalizes the library sizes of a dataset (this could be understood as normalization), creating pseudocounts that represents quantile-adjusted counts as if all samples had the same library size, while using an iterative procedure to estimate the dispersion parameter.

Usage

```
estimateDispIter(object,N=exp(mean(log(object$samples$lib.size))),prior.n=10,com
```

Arguments

`object`	object of class `DGEList` containing (at least) the elements `counts` (table of raw counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
`N`	numeric scalar giving the library size to which to normalize; default is the geometric mean of the original library sizes
`prior.n`	numeric scalar; the smoothing parameter that indicates the weight to give to the common likelihood compared to the individual tag's likelihood; default value of `10` means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion
`common.disp`	logical, if `TRUE` then the common dispersion estimate is used for all tags/genes, otherwise tag/genewise dispersion parameters are estimated; default `FALSE`
`null.hypothesis`	logical, whether to calculate the means and percentile under the null hypothesis; default is `FALSE`
`n.iter`	number of iterations in estimating the dispersion parameter
`disp.init`	numeric vector or scalar giving initialized value(s) of the dispersion parameter, can be a common value or tag/genewise values; if `NULL`, then the common value on Poisson-adjusted counts is used
`tol`	numeric scalar, tolerance in estimating the dispersion parameter
`verbose`	logical, whether to write comments, default `TRUE`

Value

list containing the following elements.

`dispersion`	numeric vector giving the estimate of the dispersion parameter for each tag/gene
`pseudo`	numeric matrix of quantile-adjusted pseudocounts
`conc`	list containing the estimates of the concentration of each tag in the underlying sample; `conc$p.common` gives estimates under the null hypothesis of no difference between groups; `conc$p.group` gives the estimate of the concentration for each tag within each group; concentration is a measure of abundance and thus expression level for the tags

N	numeric scalar, the common library size to which the counts have been adjusted
mu	numeric matrix of means that the quantile adjustment is based on

Author(s)

Mark Robinson, Davis McCarthy

See Also

The use of estimateCommonDisp and estimateTagwiseDisp are preferred for the calculation of the common dispersion and tagwise dispersion estimates, respectively.

Examples

```
set.seed(0)
y<-matrix(rnbinom(40,size=1,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
disp.out<-estimateDispIter(d,prior.n=10)
```

---

estimatePs *Estimate Expression Levels*

Description

Estimate expression levels (i.e. proportion of all sample mRNA corresponding to each tag; or, concentration of mRNA for each tag in sample mRNA) using maximum likelihood with dispersion parameter fixed based on the negative binomial model for each tag/gene and sample group. Expression proportions are used to determine overall abundance of each tag/gene and differential expression of tags/genes between groups.

Usage

```
estimatePs(object, r, tol = 1e-10, maxit = 30)
```

Arguments

object	list containing (at least) the elements counts (table of counts), group (vector or factor indicating group) and lib.size (numeric vector of library sizes)
r	numeric vector providing the size parameter of negative binomial model (size = 1/phi where phi is the dispersion parameter in the NB model)
tol	numeric scalar, tolerance between iterations
maxit	positive integer scalar, maximum number of iterations

Details

The Newton-Raphson method is used to calculate iteratively the maximum likelihood estimate of the expression level (i.e. concentration of mRNA for a particular tag in the sample mRNA) for each tag/gene.

Value

A list with elements:

conc.common	numeric vector giving overall proportion/concentration for each tag
conc.group	numeric matrix with columns giving estimates of tag/gene concentrations (proportion of total RNA for that group that that particular tag/gene contributes) for different groups)

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
set.seed(0)
y<-matrix(rnbinom(40,size=1,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
conc<-estimatePs(d,r=1)
```

---

estimateSmoothing *Estimate the Prior Weight*

Description

Estimate the prior weight, prior.n, using an approximate empirical Bayes rule given the estimate of the common dispersion. The prior weight determines how much smoothing takes place to squeeze tag/genewise estimates of the dispersion closer to the estimate of the common dispersion.

Usage

```
estimateSmoothing(object,verbose=TRUE)
```

Arguments

object	DGEList object, output of estimateCommonDisp
verbose	logical, whether to write comments, default true

Details

We are not recommending this function for routine use at the moment, as it has given unexpected results on some deep-sequenced data sets. It should be considered experimental. We are instead recommending that prior.n be chosen by the user. Values in the range 10-50 give good results in practice.

Value

estimateSmoothing produces an object of class DGEList with the following components.

prior.n	scalar; estimate of the prior weight, i.e. the smoothing parameter that indicates the weight to put on the common likelihood compared to the individual tag's likelihood; prior.n of 10 means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
d<-estimateCommonDisp(d)
prior.n<-estimateSmoothing(d)
```

---

estimateTagwiseDisp

Maximizes the Negative Binomial Weighted Conditional Likelihood

---

Description

Maximizes the negative binomial weighted likelihood (a weighted version using the common likelihood given weight according the the smoothing parameter prior.n and the individual tag/gene likelihood) for each tag from the pseudocounts provided (i.e. assuming library sizes are equal), to give an estimate of the dispersion parameter for each tag (i.e. tagwise dispersion estimation).

Usage

```
estimateTagwiseDisp(object, prior.n=10, tol=1e-06, grid=TRUE, grid.length=200, v
```

Arguments

`object`	a `DGEList` object containing (at least) the elements `counts` (table of raw counts), `group` (factor indicating group), `lib.size` (numeric vector of library sizes) and `pseudo,alt` (numeric matrix of quantile-adjusted pseudocounts calculated under the alternative hypothesis of a true difference between groups; recommended to use the `DGEList` object provided as the output of `estimateCommonDisp`
`prior.n`	numeric scalar, smoothing parameter that indicates the weight to give to the common likelihood compared to the individual tag's likelihood; default `10` means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion
`tol`	numeric scalar, if `grid=FALSE`, tolerance for Newton-Rhapson iterations
`grid`	logical, whether to use a grid search (default = `TRUE`); if `FALSE`, uses `optimize`, but this is very slow if there is a large number of tags/genes to be analysed (i.e. more than 5000)
`grid.length`	if `grid=TRUE`, the number of points at which the likelihood is evaluated for each tag, so larger values improve the accuracy of the dispersion estimates; default `1000`
`verbose`	logical, whether to write comments, default `TRUE`

Value

`estimateSmoothing` produces an object of class `DGEList` with the following components.

`common.dispersion`
: estimate of the common dispersion; the value for `phi`, the dispersion parameter in the NB model, that maximizes the negative binomial common likelihood on the `phi` scale

`prior.n`
: estimate of the prior weight, i.e. the smoothing parameter that indicates the weight to put on the common likelihood compared to the individual tag's likelihood; prior.n of 10 means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion

`tagwise.dispersion`
: tag- or gene-wise estimates of the dispersion parameter

`counts`
: table of unadjusted counts

`group`
: vector indicating the group to which each library belongs

`lib.size`
: vector containing the unadjusted size of each library

`pseudo.altn`
: table of adjusted counts; quantile-to-quantile method (see `q2qnbinom`) used to adjust the raw counts so that library sizes are equal; adjustment here done under the alternative hypothesis that there is a true difference between groups

`conc`
: list containing the estimates of the concentration of each tag in the underlying sample; `conc$p.common` gives estimates under the null hypothesis of no difference between groups; `conc$p.group` gives the estimate of the concentration for each tag within each group; concentration is a measure of abundance and thus expression level for the tags

`common.lib.size`
: the common library size to which the count libraries have been adjusted

Author(s)

Mark Robinson, Davis McCarthy

References

Robinson MD and Smyth GK (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics* 23, 2881-2887

See Also

`estimateCommonDisp` estimates a common value for the dispersion parameter for all tags/genes - should generally be run before `estimateTagwiseDisp`.

Examples

```
y<-matrix(rnbinom(1000,mu=10,size=2),ncol=4)
d<-DGEList(counts=y,group=c(1,1,2,2),lib.size=c(1000:1003))
d<-estimateCommonDisp(d)
tgwdisp<-estimateTagwiseDisp(d, prior.n=10)
```

exactTest *An Exact Test for Differences between Two Negative Binomial Groups*

Description

Carry out an exact test for differences between two negative binomial groups, based on conditioning on sums of (quantile-adjusted pseudo-)counts; calculations performed by `exactTest.matrix`

Usage

```
exactTest(object,pair=NULL,dispersion=NULL,common.disp=TRUE)
exactTest.matrix(y1,y2,mus,r,allZeros=rep(FALSE,nrow(y1)))
```

Arguments

`object`	a `DGEList` object, output of `estimateCommonDisp`, on which to compute Fisher-like exact statistics for the pair of groups specified.
`pair`	vector of length two, either numeric or character, providing the pair of groups to be compared; if a character vector, then should be the names of two groups (e.g. two levels of `object$samples$group`); if numeric, then groups to be compared are chosen by finding the levels of `object$samples$group` corresponding to those numeric values and using those levels as the groups to be compared; if `NULL`, then first two levels of `object$samples$group` (a factor) are used.
`dispersion`	optional vector either of length 1 or the same length as the number of tags. If not `NULL` (default), then the supplied value(s) will be used as the dispersion parameter for calculating p-values for differential expression. If `NULL`, then either the common or tagwise dispersion estimates from the `DGEList` object will be used, according to the value of `common.disp`. If `dispersion` is zero, then p-values are equivalent to exact Poisson rather than NB p-values.
`common.disp`	logical, if `TRUE`, then testing carried out using common dispersion for each tag/gene, if `FALSE` then tag-wise estimates of the dispersion parameter are used; default `TRUE`.
`y1`	numeric matrix of counts for one of the two given experimental groups to be tested for differences. Libraries are assumed to be equal in size - e.g. adjusted pseudocounts from the output of `equalizeLibSizes`.
`y2`	numeric matrix of counts for one of the two given experimental groups to be tested for differences. Libraries are assumed to be equal in size - e.g. adjusted pseudocounts from the output of `equalizeLibSizes`. Must have the same number of rows as `y1`.
`mus`	vector of count means for each tag/transcript under the null hypothesis (of no difference between groups)
`r`	vector of negative binomial `size` parameter values (`size = 1/phi` where `phi` is the dispersion parameter in the NB model); if `r` is of length 1, then a common value of the dispersion is used for all transcripts, otherwise, must be a vector with length equal to the number of rows of `y1` and `y2`. If you want to run a Poisson test, set r very large (e.g. 1000)
`allZeros`	logical vector indicating for each tag whether it has zero counts in each library (`TRUE`) or not (`FALSE`), with the default being not to remove any tags.

Details

For each transcript, conditioning on the total sum of counts within each group and the total sum of counts across all groups allows us to construct an exact test for differences between two group. The conditional distribution for the sum of counts in a group is known (given the values for the mean counts, `mus`, and the dispersion parameter, 1/`r`), exact p-values can be computed by summing over all sums of counts that have a probability less than the probability under the null hypothesis of the observed sum of counts.

`exactTest.matrix` is the function that actually computes the exact p-values. `exactTest` is intended to have a more object-orientated flavor as it produces objects containing all the necessary components for downstream analysis.

Value

`exactTest` produces an object of class `deDGEList` containing the following elements.

`table`	a data frame containing the elements `logConc`, the log-average concentration/abundance for each tag in the two groups being compared, `logFC`, the log-abundance ratio, i.e. fold change, for each tag in the two groups being compared, `p.value`, exact p-value for differential expression using the NB model
`comparison`	a vector giving the names of the two groups being compared
`genes`	a data frame containing information about each transcript; taken from `object` and can be `NULL`

`exactTest.matrix` produces a numeric vector of exact p-values with length equal to the number of transcripts, taken to be the number of rows of `y1`.

Author(s)

Mark Robinson, Davis McCarthy

References

Robinson MD and Smyth GK (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9, 321-332

See Also

Computing p-values for differential expression for each transcript between two (only) digital gene expression libraries can also be done using the `sage.test` function in the `statmod` package.

Examples

```
# generate raw counts from NB, create list object
y<-matrix(rnbinom(80,size=1,mu=10),nrow=20)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
rownames(d$counts)<-paste("tagno",1:nrow(d$counts),sep=".")

# estimate common dispersion and find differences in expression
d<-estimateCommonDisp(d)
de<-exactTest(d)

# example using exactTest.matrix directly
y<-matrix(rnbinom(20,mu=10,size=1.5),nrow=5)
group<-factor(c(1,1,2,2))
```

```
y<-splitIntoGroupsPseudo(y,group,pair=c(1,2))
mus<-rep(10,5)
f<-exactTest.matrix(y$y1,y$y2,mus,r=1.5,allZeros=rep(FALSE,length=nrow(y$y1)))
```

`findMaxD2` *Maximizes the Negative Binomial Likelihood*

Description

Maximizes the negative binomial likelihood (a weighted version using the common likelihood given weight alpha) for each tag

Usage

```
findMaxD2(object, alpha = 0.5, grid = TRUE, tol = 1e-05, n.iter = 10, grid.lengt
```

Arguments

`object`	list containing the raw counts with elements `counts` (table of counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
`alpha`	weight given to common likelihood, set to 0 for individual estimates or large (e.g. 100) for common likelihood
`grid`	logical, whether to use a grid search (default = `TRUE`); if `FALSE` use Newton-Rhapson steps
`tol`	if `grid=FALSE`, tolerance for Newton-Rhapson iterations
`n.iter`	if `grid=FALSE`, number of Newton-Rhapson iterations
`grid.length`	length of the grid over which to maximize; default `200`

Details

An older function, no longer called by the functions recommended to carry out analysis of DGE data, namely `estimateCommonDisp`, `estimateTagwiseDisp` and `exactTest`.

Value

vector of the values of delta that maximize the negative binomial likelihood for each tag (where delta = `phi / (phi+1)` and `phi` is the overdispersion parameter)

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y<-matrix(rnbinom(1000,mu=10,size=2),ncol=4)
d<-DGEList(counts=y,group=c(1,1,2,2),lib.size=c(1000:1003))
cml1<-findMaxD2(d,alpha=10)
cml2<-findMaxD2(d,alpha=0)
```

getCounts *Extract Table of Counts from DGEList Object*

Description

Returns the counts slot of a DGEList object

Usage

```
getCounts(object)
```

Arguments

object	DGEList object containing (at least) the elements counts (table of raw counts), group (factor indicating group) and lib.size (numeric vector of library sizes)

Value

getCounts returns a matrix of counts (presumably integers)

Author(s)

Mark Robinson, Davis McCarthy

See Also

DGEList for more information about the DGEList class.

Examples

```
# generate raw counts from NB, create list object
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
# should be 5x4
print(dim(getCounts(d)))
```

interpolateHelper *Quantile Adjustment Interpolator*

Description

Helper function to interpolate the quantile function. This is the function that actually generates the pseudocounts required by quantileAdjust to adjust (normalise) the library sizes and estimate the dispersion parameter. Given fixed values of the estimated mean (mu) and proportion in the library (p) for each tag, as well as a fixed (tagwise or common) value for the dispersion parameter (r), the function interpolates linearly the quantiles used as pseudocounts. If any value of r is infinite (corresponding to phi=0, the dispersion parameter for the negative binomial model), then a Poisson model is used, as setting phi=0 in the negative binomial model is equivalent to using a Poisson model. Otherwise, quantiles are calculated from the negative binomial distribution.

Usage

```
interpolateHelper(mu, p, r, count.max, verbose=TRUE)
```

Arguments

mu	matrix of means
p	matrix of percentiles
r	scalar, vector or matrix of size parameters
count.max	vector of maximum counts for all tags
verbose	whether to write comments, default true

Details

An older function, no longer called by the functions recommended to carry out analysis of DGE data, namely estimateCommonDisp, estimateTagwiseDisp and exactTest.

Value

numeric matrix of quantile-adjusted pseudocounts

Author(s)

Mark Robinson, Davis McCarthy

See Also

The function q2qnbinom now performs the quantile-adjustment to equalize library sizes and generate pseudocounts - this newer function is faster and more accurate.

Examples

```
y<-matrix(rnbinom(10000,size=2,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000,1010),2))
conc<-estimatePs(d,r=2)
N<-exp(mean(log(d$samples$lib.size)))
perc<-pnbinom(d$counts-1,size=2,mu=outer(conc$conc.common,d$samples$lib.size))+dnbinom(d$
maxcounts<-apply(d$counts,1,max)
pseudo<-interpolateHelper(outer(conc$conc.common,rep(N,4)),perc,r=2,maxcounts)
```

---

logLikDerP *Log-Likelihood for Proportion*

Description

Log-likelihood and derivatives for the proportion parameter (i,e, expression level) of negative binomial (mean = library size * proportion)

Usage

```
logLikDerP(p, y, lib.size, r, der = 0)
```

Arguments

p	vector of proportion parameters to be evaluated
y	matrix of counts
lib.size	vector of library sizes
r	size parameter of negative binomial distribution
der	derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)

Value

vector of the likelihood or specified derivative evaluations for each tag/gene

Author(s)

Mark Robinson, Davis McCarthy

See Also

estimatePs calls logLikDerP as part of the procedure for estimating the expression level(s) of each tag.

Examples

```
y<-matrix(rnbinom(20,size=1.5,mu=10),nrow=5)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))

this.p<-rowMeans( y/ outer(rep(1,nrow(y)),d$samples$lib.size) )
d1p<-logLikDerP(this.p,y,d$samples$lib.size,r=1.5,der=1)
```

---

maPlot — *Plots Log-Fold Change versus Log-Concentration (or, M versus A) for Count Data*

Description

To represent counts that were low (e.g. zero in 1 library and non-zero in the other) in one of the two conditions, a 'smear' of points at low A value is presented.

Usage

```
maPlot(x, y, normalize=FALSE, smearWidth = 1, col = NULL, allCol = "black", lowC
```

Arguments

x	vector of counts or concentrations (group 1)
y	vector of counts or concentrations (group 2)
normalize	logical, whether to divide x and y vectors by their sum
smearWidth	scalar, width of the smear
col	vector of colours for the points (if NULL, uses allCol and lowCol)
allCol	colour of the non-smeared points

lowCol	colour of the smeared points
deCol	colour of the DE (differentially expressed) points
de.tags	indices for tags identified as being differentially expressed; use exactTest to identify DE genes
...	further arguments passed on to plot

Details

The points to be smeared are identified as being equal to the minimum in one of the two groups. The smear is created by using random uniform numbers of width smearWidth to the left of the minimum A value.

Value

a plot to the current device

Author(s)

Mark Robinson

See Also

plotSmear

Examples

```
y <- matrix(rnbinom(10000,mu=5,size=2),ncol=4)
maPlot(y[,1], y[,2])
```

---

plotMDS.dge *Multidimensional scaling plot of SAGE data*

Description

Plot the sample relations based on Multidimensional Scaling.

Usage

```
plotMDS.dge(x, top=500, col=NULL, cex=1, dim.plot=c(1,2), ndim=max(dim.plot),...
```

Arguments

x	any matrix or DGEList object.
top	number of top genes used to calculate pairwise distances.
col	numeric or character vector of colors for the plotting characters.
cex	numeric vector of plot symbol expansions.
dim.plot	which two dimensions should be plotted, numeric vector of length two.
ndim	number of dimensions in which data is to be represented
...	any other arguments are passed to plot.

Details

This function is a variation on the usual multdimensional scaling (or principle coordinate) plot, in that a distance measure particularly appropriate for the digital gene expression (DGE) context is used. The distance between each pair of samples (columns) is the square root of the common dispersion for the top `top` genes which best distinguish that pair of samples. These top `top` genes are selected according to the tagwise dispersion of all the samples.

See `text` for possible values for `col` and `cex`.

Value

A plot is created on the current graphics device.

Author(s)

Yunshun Chen and Gordon Smyth

Examples

```
# Simulate DGE data for 1000 genes(tags) and 6 samples.
# Samples are in two groups
# First 300 genes are differentially expressed in second group

x <- 10*runif(1000)
counts <- rnbinom(6000, size = 5, mu = x)
m <- matrix(counts, 1000, 6)
rownames(m) <- paste("Gene",1:1000)
m[1:300,4:6] <- m[1:300,4:6] + 10
plotMDS.dge(m)

# Indexes of samples are plotted.
plotMDS.dge(m,  col=c(rep("black",3), rep("red",3)) )
```

---

plotSmear — *Plots log-Fold Change versus log-Concentration (or, M versus A) for Count Data*

Description

Both of these functions plot the log-fold change (i.e. the log of the ratio of expression levels for each tag between two experimental groups) against the log-concentration (i.e. the overall average expression level for each tag across the two groups). To represent counts that were low (e.g. zero in 1 library and non-zero in the other) in one of the two conditions, a 'smear' of points at low A value is presented in `plotSmear`.

Usage

```
plotSmear(object, pair = NULL, de.tags=NULL, xlab = "logConc", ylab = "logFC", p
plotFC(de.object,xlab="logConc",ylab="logFC",ylim=NULL,pch=19,cex=0.2,...)
```

Arguments

object	DGEList object to plot data from (uses $conc element)
de.object	deDGEList object , as output from exactTest
pair	pair of experimental conditions to plot (if NULL, the first two conditions are used)
de.tags	rownames for tags identified as being differentially expressed; use exactTest to identify DE genes
xlab	x-label of plot
ylab	y-label of plot
pch	scalar or vector giving the character(s) to be used in the plot; default value of 19 gives a round point.
cex	character expansion factor, numerical value giving the amount by which plotting text and symbols should be magnified relative to the default; default cex=0.2 to make the plotted points smaller
smearWidth	width of the smear
panel.first	an expression to be evaluated after the plot axes are set up but before any plotting takes place; the default grid() draws a background grid to aid interpretation of the plot
ylim	vector of length two giving limits on y-axis, if left at NULL, scaled to be symmetric about 0
...	further arguments passed on to plot

Details

While both functions do essentially the same thing, plotSmear is a more sophisticated and superior way to produce an 'MA plot'. plotFC is an earlier and rawer version of the plotting function and has difficulty dealing with tags which have a total count of zero for one of the groups—this issue is resolved in plotSmear by adding the 'smear' of points at low A value. The points to be smeared are identified as being equal to the minimum estimated concentration in one of the two groups. The smear is created by using random uniform numbers of width smearWidth to the left of the minimum A. plotSmear also allows easy highlighting of differentially expressed (DE) tags, and the use of plotSmear is strongly recommended over plotFC.

Value

A plot to the current device

Author(s)

Mark Robinson, Davis McCarthy

See Also

maPlot

Examples

```
y <- matrix(rnbinom(10000,mu=5,size=2),ncol=4)
d <- DGEList(counts=y, group=rep(1:2,each=2), lib.size=colSums(y))
rownames(d$counts) <- paste("tag",1:nrow(d$counts),sep=".")
d <- estimateCommonDisp(d)
plotSmear(d)

# find differential expression
de<-exactTest(d)

# plot it
plotFC(de)
# highlighting the top 500 most DE tags
de.tags <- rownames(topTags(de, n=500)$table)
plotSmear(d, de.tags=de.tags)
```

---

q2qnbinom *Quantile to Quantile Mapping between Negative-Binomial Distributions*

Description

Approximate quantile to quantile mapping between negative-binomial distributions with the same dispersion but different means. The Poisson distribution is a special case.

Usage

```
q2qpois(x, input.mean, output.mean)
q2qnbinom(x, input.mean, output.mean, dispersion=0)
```

Arguments

x	numeric matrix of unadjusted count data from a DGEList object
input.mean	numeric matrix of estimated mean counts for tags/genes in unadjusted libraries
output.mean	numeric matrix of estimated mean counts for tags/genes in adjusted (equalized) libraries, the same for all tags/genes in a particular group, different between groups
dispersion	numeric scalar, vector or matrix of dispersion parameters

Details

This function finds the quantile with the same left and right tail probabilities relative to the output mean as x has relative to the input mean. q2qpois is equivalent to q2qnbinom with dispersion=0.

This is the function that actually generates the pseudodata for equalizeLibSizes and required by estimateCommonDisp to adjust (normalize) the library sizes and estimate the dispersion parameter. The function takes fixed values of the estimated mean for the unadjusted libraries (input.mean) and the estimated mean for the equalized libraries (output.mean) for each tag, as well as a fixed (tagwise or common) value for the dispersion parameter (phi).

The function calculates the percentiles that the counts in the unadjusted library represent for the normal and gamma distributions with mean and variance defined by the negative binomial rules: mean=`input.mean` and variance=`input.mean*(1+dispersion*input.mean)`. The percentiles are then used to obtain quantiles from the normal and gamma distributions respectively, with mean and variance now defined as above but using `output.mean` instead of `input.mean`. The function then returns as the pseudodata, i.e., equalized libraries, the arithmetic mean of the quantiles for the normal and the gamma distributions. As the actual negative binomial distribution is not used, we refer to this as a "poor man's" NB quantile adjustment function, but it has the advantage of not producing Inf values for percentiles or quantiles as occurs using the equivalent NB functions. If, for any tag, the dispersion parameter for the negative binomial model is 0, then it is equivalent to using a Poisson model. Lower tails of distributions are used where required to ensure accuracy.

Value

numeric matrix of the same size as `x` with quantile-adjusted pseudodata

Author(s)

Gordon Smyth

Examples

```
y<-matrix(rnbinom(10000,size=2,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000,1010),2))
conc<-estimatePs(d,r=2)
N<-exp(mean(log(d$samples$lib.size)))
in.mean<-matrix(0,nrow=nrow(d$counts),ncol=ncol(d$counts))
out.mean<-matrix(0,nrow=nrow(d$counts),ncol=ncol(d$counts))
for(i in 1:2) {
in.mean[,d$samples$group==i]<-outer(conc$conc.group[,i],d$samples$lib.size[d$samples$grou
out.mean[,d$samples$group==i]<-outer(conc$conc.group[,i],rep(N,sum(d$samples$group==i)))
}
pseudo<-q2qnbinom(d$counts, input.mean=in.mean, output.mean=out.mean, dispersion=0.5)
```

---

quantileAdjust *Normalizes a Dataset by Using a Quantile Adjustment*

Description

The function adjusts (you might say normalizes) a dataset, creating pseudocounts that represents quantile-adjusted counts as if all samples had the same library size, while estimating the dispersion parameter.

Usage

```
quantileAdjust(object, N = exp(mean(log(object$samples$lib.size))), alpha = 0, n
```

Arguments

object	list containing the raw counts with elements counts (table of counts), group (factor indicating group) and lib.size (numeric vector of library sizes)
N	library size to normalize to; default is the geometric mean of the original library sizes
alpha	weight to put on the individual tag's likelihood
null.hypothesis	logical, whether to calculate the means and percentile under the null hypothesis; default is FALSE
n.iter	number of iterations in estimating the size parameter
r.init	initialized value of the size parameter; if NULL, then the common value on unadjusted counts is used
tol	tolerance in estimating the size parameter
verbose	whether to write comments, default true

Details

An older function, no longer called by the functions recommended to carry out analysis of DGE data, namely estimateCommonDisp, estimateTagwiseDisp and exactTest. No longer recommended for use.

Value

list containing several elements used in downstream function calls.

r	is the dispersion estimate
pseudo	is the quantile-adjusted pseudocounts
ps	is a list containing the abundance estimates
N	is the common library size
p	percentiles on which the quantile is based
mu	means on which the quantile is based

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
set.seed(0)
y<-matrix(rnbinom(40,size=1,mu=10),ncol=4)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
qA<-quantileAdjust(d,alpha=100)
```

readDGE *Read and Merge a Set of Files Containing DGE Data*

Description

Reads and merges a set of text files containing digital gene expression data.

Usage

```
readDGE(files,path=NULL,columns=c(1,2),group=NULL,...)
```

Arguments

files	character vector of filenames, or alternatively a data.frame with a column containing the file names of the files containing the libraries of counts and, optionally, columns containing the group to which each library belongs, descriptions of the other samples and other information.
path	character string giving the directory containing the files. The default is the current working directory.
columns	numeric vector stating which two columns contain the tag names and counts, respectively
group	vector, or preferably a factor, indicating the experimental group to which each library belongs. If group is not NULL, then this argument overrides any group information included in the files argument.
...	other are passed to read.delim

Details

Each file is assumed to contained digital gene expression data for one sample (or library), with transcript identifiers in the first column and counts in the second column. Transcript identifiers are assumed to be unique and not repeated in any one file. By default, the files are assumed to be tab-delimited and to contain column headings. The function forms the union of all transcripts and creates one big table with zeros where necessary.

Value

DGEList object

Author(s)

Mark Robinson and Gordon Smyth

See Also

DGEList provides more information about the DGEList class and the function DGEList, which can also be used to construct a DGEList object, if readDGE is not required to read in and construct a table of counts from separate files.

Examples

```
#  Read all .txt files from current working directory

## Not run: files <- dir(pattern="*\\.txt$")
RG <- readDGE(files)
## End(Not run)
```

`splitIntoGroups` *Split the Counts or Pseudocounts from a DGEList Object According To Group*

Description

Split the counts from a DGEList object according to group, creating a list where each element consists of a numeric matrix of counts for a particular experimental group. Given a pair of groups, split pseudocounts for these groups, creating a list where each element is a matrix of pseudocounts for a particular gourp.

Usage

```
splitIntoGroups(object)
splitIntoGroupsPseudo(pseudo, group, pair)
```

Arguments

`object`	`DGEList`, object containing (at least) the elements `counts` (table of raw counts), `group` (factor indicating group) and `lib.size` (numeric vector of library sizes)
`pseudo`	numeric matrix of quantile-adjusted pseudocounts to be split
`group`	factor indicating group to which libraries/samples (i.e. columns of `pseudo` belong; must be same length as ncol(pseudo)
`pair`	vector of length two stating pair of groups to be split for the pseudocounts

Value

`splitIntoGroups` outputs a list in which each element is a matrix of count counts for an individual group. `splitIntoGroupsPseudo` outputs a list with two elements, in which each element is a numeric matrix of (pseudo-)count data for one of the groups specified.

Author(s)

Davis McCarthy

Examples

```
# generate raw counts from NB, create list object
y<-matrix(rnbinom(80,size=1,mu=10),nrow=20)
d<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
rownames(d$counts)<-paste("tagno",1:nrow(d$counts),sep=".")
z1<-splitIntoGroups(d)

z2<-splitIntoGroupsPseudo(d$counts,d$group,pair=c(1,2))
```

subsetting *Subset DGEList Objects*

Description

Extract a subset of a DGEList object.

Usage

```
## S3 method for class 'DGEList':
object[i, j, ...]
```

Arguments

object	object of class DGEList
i,j	elements to extract. i subsets the tags or genes while j subsets the libraries
...	not used

Details

i,j may take any values acceptable for the matrix components of object. See the Extract help entry for more details on subsetting matrices.

Value

An object of class DGEList holding data from the specified subset of tags/genes and libraries.

Author(s)

Davis McCarthy, Gordon Smyth

See Also

Extract in the base package.

Examples

```
d <- matrix(rnbinom(8,size=1,mu=10),4,2)
rownames(d) <- c("a","b","c","d")
colnames(d) <- c("A","B")
d <- new("DGEList",list(counts=d,group=factor(c("A","B"))))
d[1:2,]
d[1:2,2]
d[,2]
```

tau2.0.objective *Objective Function for Tau2*

Description

Objective function for tau2, which is used in the approximate empirical Bayes rule which determines how much to squeeze the dispersion parameters towards the common value. Tau2 is analogous to the prior variance for each tag/gene in an hierarchical model for the estimators of the tag/genewise dispersion parameter, and must be estimated in order to select the smoothing parameter as an approximate EB rule.

Usage

```
tau2.0.objective(tau2.0, info.g, score.g)
```

Arguments

tau2.0	scalar, value for tau2
info.g	numeric vector, observed information for each tag/gene
score.g	scalar, observed score (first derivative of log-likelihood) over all tags/genes

Value

scalar, value of objective function at tau2.0

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
x<-DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(1000:1001,each=2))
scores <- condLogLikDerDelta(y, delta=0.5, der = 1, doSum = TRUE)
q2q.out<-equalizeLibSizes(x,disp=1,null.hypothesis=TRUE)
exp.inf<-approx.expected.info(x,d=0.5,q2q.out$pseudo)
sigma2.0.est<-optimize(tau2.0.objective,c(0,500),info.g=exp.inf,score.g=scores)$min
```

topTags *Table of the Top Differentially Expressed Tags*

Description

Extracts the top DE tags in a data frame for a given pair of groups, ranked by p-value or absolute log-fold change.

Usage

```
topTags(object, n=10, adjust.method="BH", sort.by="p.value")
```

Arguments

object	a deDGEList object, as output from exactTest, containing the elements: table: data frame containing the log-concentration (i.e. expression level), the log-fold change in expression between the two groups/conditions and the exact p-value for differential expression, for each tag. comparison: vector giving the two experimental groups/conditions being compared. genes: data frame containing information about each transcript (can be NULL).
n	scalar, number of tags to display/return
adjust.method	character string stating the method used to adjust p-values for multiple testing, passed on to p.adjust
sort.by	character string, indicating whether tags should be sorted by p-value ("p.value") or absolute log-fold change ("logFC"); default is to sort by p-value.

Value

an object of class TopTags containing the following elements for the top n most differentially expressed tags as determined by sort.by. There is a show method for this class.

table	a data frame containing the elements logConc, the log-average concentration/abundance for each tag in the two groups being compared, logFC, the log-abundance ratio, i.e. fold change, for each tag in the two groups being compared, p.value, exact p-value for differential expression using the NB model, adj.p.val, the p-value adjusted for multiple testing as found using p.adjust using the method specified
comparison	a vector giving the names of the two groups being compared

Author(s)

Mark Robinson, Davis McCarthy, Gordon Smyth

References

Robinson MD, Smyth GK. 'Small-sample estimation of negative binomial dispersion, with applications to SAGE data.' Biostatistics. 2008 Apr;9(2):321-32.

Robinson MD, Smyth GK. 'Moderated statistical tests for assessing differences in tag abundance.' Bioinformatics. 2007 Nov 1;23(21):2881-7.

See Also

exactTest, p.adjust.

Analogous to topTable in the limma package.

Examples

```
# generate raw counts from NB, create list object
y <- matrix(rnbinom(80,size=1,mu=10),nrow=20)
d <- DGEList(counts=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
rownames(d$counts) <- paste("tag",1:nrow(d$counts),sep=".")

# estimate common dispersion and find differences in expression
```

```
d<-estimateCommonDisp(d)
de<-exactTest(d)

# look at top 10
topTags(de)
# Can specify how many tags to view
tp <- topTags(de, n=15)
# Here we view top 15
tp
# Or order by fold change instead
topTags(de,sort.by="logFC")
```

`weightedCondLogLikDerDelta` *Weighted Conditional Log-Likelihood in Terms of Delta*

Description

Weighted conditional log-likelihood parameterized in terms of delta (`phi / (phi+1)`) for a given tag/gene - maximized to find the smoothed (moderated) estimate of the dispersion parameter

Usage

```
weightedCondLogLikDerDelta(y, delta, tag, prior.n=10, ntags=nrow(y[[1]]), der=0,
```

Arguments

`y`	list with elements comprising the matrices of count data (or pseudocounts) for the different groups
`delta`	delta (`phi / (phi+1)`)parameter of negative binomial
`tag`	tag/gene at which the weighted conditional log-likelihood is evaluated
`prior.n`	smoothing paramter that indicates the weight to put on the common likelihood compared to the individual tag's likelihood; default `10` means that the common likelihood is given 10 times the weight of the individual tag/gene's likelihood in the estimation of the tag/genewise dispersion
`ntags`	numeric scalar number of tags/genes in the dataset to be analysed
`der`	derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)
`doSum`	logical, whether to sum over samples or not (default `FALSE`

Details

This function computes the weighted conditional log-likelihood for a given tag, parameterized in terms of delta. The value of delta that maximizes the weighted conditional log-likelihood is converted back to the `phi` scale, and this value is the estimate of the smoothed (moderated) dispersion parameter for that particular tag. The delta scale for convenience (delta is bounded between 0 and 1).

Value

numeric scalar of function/derivative evaluated for the given tag/gene and delta

Author(s)

Mark Robinson, Davis McCarthy

Examples

```
counts<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(counts=counts,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
y<-splitIntoGroups(d)
ll1<-weightedCondLogLikDerDelta(y,delta=0.5,tag=1,prior.n=10,der=0)
ll2<-weightedCondLogLikDerDelta(y,delta=0.5,tag=1,prior.n=10,der=1)
```

Index