edgeR
April 19, 2009

DGEList-class  Digital Gene Expression data - class

Description
A simple list-based class for storing read counts from digital gene expression technologies.

Slots/List Components
Objects of this class contain the following list components:

data: numeric matrix containing the read counts.
lib.size: numeric vector containing the total number of reads for each library (column of code).
group: vector giving the experimental group/condition.

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

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.
alpha.approxeb

score: numeric scalar (likelihood) score.
inos: numeric vector containing the (likelihood) information.
exact: list containing info with respect to quantile adjustment.

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

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 data with elements data (table of counts), group (vector indicating group) and lib.size (vector of library sizes)
verbose whether to write comments, default true

Value

EBList object with elements p (overall proportion), p1 (estimates for first group), p2 (estimates for second group)

Author(s)

Mark Robinson

Examples

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

Approximate of 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**

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

**Arguments**

- `object` DGEList object containing the raw data with elements `data` (table of counts), `group` (vector indicating group) and `lib.size` (vector of library sizes)
- `d` delta parameter for negative binomial - phi/(phi+1)
- `qA` list from output of `quantileAdjust`
- `robust` logical on whether to use a robust fit, default `FALSE`

**Value**

vector of Fisher information approximates (with length same as the number of rows of the original data)

**Author(s)**

Mark Robinson

**Examples**

```r
set.seed(0)
y<-matrix(rnbinom(40,size=1,mu=10),ncol=4)
d<-list(data=y,group=rep(1:2,each=2),lib.size=rep(c(1000:1001),2))
qA<-quantileAdjust(d,alpha=100)
exp.inf<-approx.expected.info(d,1/(1 + qA$r[1]),qA)
```

**condLogLikDerDelta** Conditional log-likelihood in terms of delta

**Description**

Conditional log-likelihood paramterized in terms of delta (phi / (phi+1))

**Usage**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>matrix with count data (or pseudo-data)</td>
</tr>
<tr>
<td>delta</td>
<td>delta ((\phi / (\phi+1)))parameter of negative binomial</td>
</tr>
<tr>
<td>grid</td>
<td>logical, whether to calculate a grid over the values of delta</td>
</tr>
<tr>
<td>der</td>
<td>derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)</td>
</tr>
<tr>
<td>doSum</td>
<td>logical, whether to sum over samples or not (default TRUE)</td>
</tr>
</tbody>
</table>

Value

vector of matrix of function/derivative evaluations

Author(s)

Mark Robinson

Examples

```r
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 Conditional log-likelihood in terms of size

Description

Conditional log-likelihood parameterized in terms of size \((1 / \phi)\)

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>list containing the raw data with elements data (table of counts), group (vector indicating group) and lib.size (vector of library sizes)</td>
</tr>
<tr>
<td>r</td>
<td>size parameter of negative binomial distribution</td>
</tr>
<tr>
<td>der</td>
<td>derivative, either 0 (the function), 1 (first derivative) or 2 (second derivative)</td>
</tr>
</tbody>
</table>

Value

vector of matrix of function/derivative evaluations

Author(s)

Mark Robinson

Examples

```r
y1<-matrix(rnbinom(10,size=1,mu=10),nrow=5)
ll2<-condLogLikDerSize(y1,r=10,der=0)
```
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 data (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

Value

dedGEList with elements lr (likelihood ratio test), r (estimates of 1/overdispersion), ps (list containing proportion estimates)

Author(s)

Mark Robinson

References


Examples

# generate raw data from NB, create list object
y<-matrix(rnbinom(20, size=1, mu=10), nrow=5)
d<-DGEList(data=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>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps</td>
<td>list containing estimates of p parameter.</td>
</tr>
<tr>
<td>r</td>
<td>numeric vector of size parameter (1/\phi) where (\phi) is negative binomial dispersion.</td>
</tr>
<tr>
<td>pseudo</td>
<td>numeric matrix with the pseudo-counts.</td>
</tr>
<tr>
<td>M</td>
<td>numeric scalar with the library size that pseudo counts are mapped to.</td>
</tr>
<tr>
<td>exact</td>
<td>numeric vector of exact p-values (unadjusted).</td>
</tr>
</tbody>
</table>

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

estimatePs  

Estimate expression proportions

Description
Estimate expression proportions (maximum likelihood with size fixed) based on negative binomial for each tag and sample group (only 2 groups implemented at this point)

Usage
estimatePs(y1, y2, lib.size1, lib.size2, r, tol = 1e-10, maxit = 30)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>matrix of counts for first group</td>
</tr>
<tr>
<td>y2</td>
<td>matrix of counts for second group</td>
</tr>
<tr>
<td>lib.size1</td>
<td>vector of library sizes for first group</td>
</tr>
<tr>
<td>lib.size2</td>
<td>vector of library sizes for second group</td>
</tr>
<tr>
<td>r</td>
<td>size parameter of negative binomial</td>
</tr>
<tr>
<td>tol</td>
<td>tolerance between iterations</td>
</tr>
<tr>
<td>maxit</td>
<td>maximum number of iterations</td>
</tr>
</tbody>
</table>
exactTestNB

Value

list with elements $p$ (overall proportion), $p_1$ (estimates for first group), $p_2$ (estimates for second group)

Author(s)

Mark Robinson

Examples

```r
y1<-matrix(rnbinom(10,size=1,mu=10),nrow=5)
y2<-matrix(rnbinom(10,size=1,mu=5),nrow=5)
ps<-estimatePs(y1,y2,c(1000,1001),c(1000,1001),r=1)
```

**exactTestNB**

An exact test for differences between two negative binomial groups

Description

An exact test for differences between two negative binomial groups

Usage

`exactTestNB(y, g, mus, r, verbose=TRUE)`

Arguments

- **y**: data (e.g. quantile adjusted pseudodata) to compute Fisher exact statistics on
- **g**: group indicator, must be same length as nrow(y)
- **mus**: vector of means under the null hypothesis (of no difference between groups)
- **r**: preset or estimated negative binomial size parameter. If you want to run a Poisson test, set $r$ very large (e.g. 1000)
- **verbose**: whether to write comments, default `true`

Value

list with elements $lr$ (likelihood ratio test), $r$ (estimates of $1/overdispersion$), $ps$ (list containing proportion estimates)

Author(s)

Mark Robinson

Examples

```r
y<-matrix(rnbinom(20,mu=10,size=1.5),nrow=5)
group<-c(1,1,2,2)
mus<-rep(10,5)
f<-exactTestNB(y,group,mus,r=1.5)
```
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(x, alpha = 0.5, grid = TRUE, tol = 1e-05, n.iter = 5, grid.length = 200)

Arguments

x  
list with elements data, lib.size and group

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 to maximize over; default 200

Value

list with elements lr (likelihood ratio test), r (estimates of 1/overdispersion), ps (list containing proportion estimates)

Author(s)

Mark Robinson

Examples

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

Extract data table from DGEList object

### Description

Returns the data slot of a DGEList object

### Usage

```r
ggetData(object)
```

### Arguments

- **object**: matrix of counts for first group

### Value

matrix of data (presumably integers)

### Author(s)

Mark Robinson

### Examples

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

interpolateHelper

Quantile Adjustment interpolator

### Description

Helper function to interpolate the quantile function

### Usage

```r
interpolateHelper(mu, p, r, d, verbose=TRUE)
```

### Arguments

- **mu**: matrix of means
- **p**: matrix of percentiles
- **r**: scalar, vector or matrix of size parameters
- **d**: original data matrix
- **verbose**: whether to write comments, default `true`
Value

matrix with quantile-adjusted pseudo data

Author(s)

Mark Robinson

Examples

```r
y<-matrix(rnbinom(10000,size=2,mu=10),ncol=4)
d<-list(data=y,group=rep(1:2,each=2),lib.size=rep(c(1000,1010),2))
ps<-estimatePs(d$data[,1:2],d$data[,3:4],d$lib.size[1:2],d$lib.size[3:4],r=2)
N<-prod(d$lib.size)^(1/ncol(d$data))
perc<-pnbinom(d$data-1,size=2,mu=outer(ps$p,d$lib.size))+dnbinom(d$data,size=2,mu=outer(ps$p,d$lib.size))/2
pseudo<-interpolateHelper(outer(ps$p,rep(N,4)),perc,r=2,d$data)
```
plotMA

MA-like plot for deDGEList objects

Description

Plots

Usage

plotMA(object,xlab="A",ylab="M",ylim=NULL,pch=19,...)

Arguments

object deDGEList object, as output from deDGE
xlab x-axis label
ylab y-axis label
ylim limits on y-axis, if left at NULL, scaled to be symmetric about 0
pch plot character
... further arguments to the plot command

Value

A plot to the current device

Author(s)

Mark Robinson

See Also

deDGE

Examples

# generate raw data from NB, create list object
y<-matrix(rnbinom(20,size=1,mu=10),nrow=5)
d<-DGEList(data=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)

# plot it
plotMA(ms)
quantileAdjust

Quantile adjust a dataset by using a quantile adjustment

Description

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

Usage

quantileAdjust(object, N = prod(object$lib.size)^(1/ncol(object$data)), alpha = 0, null.hypothesis = FALSE, n.iter = 5, r.init = NULL, tol = 0.001, verbose = TRUE)

Arguments

object  list containing the raw data with elements data (table of counts), group (vector indicating group) and lib.size (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 TRUE
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 un-adjusted data is used
tol tolerance in estimating the size parameter
verbose whether to write comments, default true

Value

list containing several elements used in downstream function calls. r is the dispersion estimate, pseudo is the quantile-adjusted pseudodata, ps is a list containing the abundance estimates, N is the common library size and p and mu are the percentiles and means, respectively that the quantile is based on

Author(s)

Mark Robinson

Examples

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

*Read a list of files containing DGE data*

**Description**

Reads a list of text files, one for each sample. Files should be tab-delimited with an identifier (could be tag sequence) as the first column and counts as the second column. The function creates one big table with 0s where necessary.

**Usage**

```r
readDGE(files,...)
```

**Arguments**

- `files` character vector of filenames
- `...` option arguments to send to `read.table`

**Value**

list with elements `data` (table of counts), `lib.size` (library sizes)

**Author(s)**

Mark Robinson

**Examples**

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

## Not run:
files <- dir(pattern="*\.txt\$")
RG <- readDGE(files,sep="\t",header=TRUE,comment.char="",stringsAsFactors=FALSE)
## End(Not run)
```

---

**tau2.0.objective**

*Objective function for tau2*

**Description**

Objective function for tau2 which is used in the rule of how much to squeeze the dispersion parameters towards the common value

**Usage**

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

- **tau2.0**: scalar, value for tau
- **info.g**: observed information for each gene
- **score.g**: observed score (first derivative of log-likelihood) for each gene

Value

- scalar, value of objective function at tau2.0

Author(s)

Mark Robinson

Examples

```r
y <- matrix(rnbinom(20, size=1, mu=10), nrow=5)
x <- list(data=y, group=rep(1:2, each=2), lib.size=rep(1000:1001, each=2))
scores <- condLogLikDerDelta(y, delta=0.5, der = 1, doSum = TRUE)
qA <- quantileAdjust(x, alpha = 10, null.hypothesis = TRUE)
exp.inf <- approx.expected.info(x, d=0.5, qA)
sigma2.0.est <- optimize(tau2.0.objective, c(0, 500), info.g = exp.inf, score.g = scores)
```

---

topTags

*Displays the top differentially expressed tags in a table*

Description

Displays/Returns the top DE tags in a data frame

Usage

```r
topTags(object, n=10, adj.method= "BH")
```

Arguments

- **object**: deDGEList, output from deDE
- **n**: number of tags to display/return
- **adj.method**: method used to adjust P-values, using p.adjust

Value

Data frame containing the relative level of expression, log fold changes, unadjusted and adjusted P-values

Author(s)

Mark Robinson
References


Examples

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

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

# look at top 10
topTags(ms)
Index

**Topic** algebra
- deDGE, 5
- exactTestNB, 7
- findMaxD2, 8
- interpolateHelper, 9
- tau2.0.objective, 13
- topTags, 14

**Topic** classes
- deDGEList-class, 6
- DGEList-class, 1
- EBList-class, 1

**Topic** file
- alpha.approxeab, 2
- approx.expected.info, 3
- condLogLikDerDelta, 3
- condLogLikDerSize, 4
- estimatePs, 6
- getData, 9
- logLikDerP, 10
- plotMA, 11
- quantileAdjust, 12
- readDGE, 13
- show, deDGEList-method (deDGEList-class), 6
- show, DGEList-method (DGEList-class), 1
- show, EBList-method (EBList-class), 1
- tau2.0.objective, 13
- topTags, 14

alpha.approxeab, 2
approx.expected.info, 3
condLogLikDerDelta, 3
condLogLikDerSize, 4
deDGE, 5
deDGEList-class, 6
DGEList (DGEList-class), 1
DGEList-class, 1
EBList-class, 1
estimatePs, 6
exactTestNB, 7
findMaxD2, 8
dataGet, 9
interpolateHelper, 9
logLikDerP, 10
plotMA, 11
plotMA, deDGEList-method (plotMA), 11
quantileAdjust, 12
readDGE, 13
show, deDGEList-method (deDGEList-class), 6
show, DGEList-method (DGEList-class), 1
show, EBList-method (EBList-class), 1
tau2.0.objective, 13
topTags, 14