Package ‘EBSeq’

January 14, 2017

Type Package

Title An R package for gene and isoform differential expression
analysis of RNA-seq data

Version 1.14.0

Date 2015-12-8

Author Ning Leng, Christina Kendzierski

Maintainer Ning Leng <lengning1@gmail.com>

Depends blockmodeling, gplots, testthat, R (>= 3.0.0)

Description Differential Expression analysis at both gene and isoform
level using RNA-seq data

License Artistic-2.0

LazyLoad yes

Collate 'MedianNorm.R' 'GetNg.R' 'beta.mom.R' 'f0.R' 'f1.R'
'Likefun.R' 'LogN.R' 'LogNMult.R' 'LikefunMulti.R' 'EBTest.R'
'GetPatterns.R' 'EBMultiTest.R' 'GetPP.R' 'PostFC.R'
'GetPPMat.R' 'GetMultiPP.R' 'GetMultiFC.R' 'PlotPostVsRawFC.R'
'crit_fun.R' 'DenNHist.R' 'GetNormalizedMat.R' 'PlotPattern.R'
'PolyFitPlot.R' 'QQP.R' 'QuantileNorm.R' 'RankNorm.R'
'GetDEResults.R'

BuildVignettes yes

biocViews StatisticalMethod, DifferentialExpression,
MultipleComparison, RNASeq, Sequencing

NeedsCompilation no

R topics documented:

EBSeq_NingLeng-package ........................................ 2
beta.mom ......................................................... 3
crit_fun ......................................................... 4
DenNHist ......................................................... 5
EBMultiTest ...................................................... 6
EBTest ........................................................... 8
f0 ............................................................... 11
f1 ............................................................... 12
GeneMat ......................................................... 13
EBSeq_NingLeng-package

EBSeq: RNA-Seq Differential Expression Analysis on both gene and isoform level

Description

In 'EBSeq_NingLeng-package,' a Negative Binomial-beta model was built to analyze the RNASeq data. We used the empirical bayes method and EM algorithm.

Details

Package: EBSeq_NingLeng
Type: Package
Version: 1.0
Date: 2011-06-13
License: What license is it under?
LazyLoad: yes

Author(s)

Ning Leng, Christina Kendziorski
Maintainer: Ning Leng <nleng@wisc.edu>

References


See Also

EBTest, EBMultiTest

Examples

```r
data(GeneMat)
GeneMat.small = GeneMat[c(1:10,511:550),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data=GeneMat.small,
Conditions=as.factor(rep(c("C1","C2"), each=5)),
sizeFactors=Sizes, maxround=5)
```
crit_fun

See Also
DenNHist, DenNHistTable

Examples

```r
#tmp = rbeta(5, 5, 100)
#param = beta.mom(tmp)
```

---

crit_fun  
*Calculate the soft threshold for a target FDR*

Description

'crit_fun' calculates the soft threshold for a target FDR.

Usage

crit_fun(PPEE, thre)

Arguments

- **PPEE**: The posterior probabilities of being EE.
- **thre**: The target FDR.

Details

Regarding a target FDR alpha, both hard threshold and soft threshold could be used. If the hard threshold is preferred, user could simply take the transcripts with PP(DE) greater than (1-alpha). Using the hard threshold, any DE transcript in the list is with FDR <= alpha.

If the soft threshold is preferred, user could take the transcripts with PP(DE) greater than crit_fun(PPEE, alpha). Using the soft threshold, the list of DE transcripts is with average FDR alpha.

Value

The adjusted FDR threshold of target FDR.

Author(s)

Ning Leng

References

Examples

```r
data(GeneMat)
GeneMat.small = GeneMat[c(1:10, 500:600),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTTest(Data = GeneMat.small,
Conditions = as.factor(rep(c("C1","C2"), each=5)),
sizeFactors = Sizes, maxround = 5)
PP = GetPPMat(EBOut)
DEfound = rownames(PP)[which(PP[,"PPDE"] >= 0.95)]
str(DEfound)
SoftThre = crit_fun(PP[,"PPEE"], 0.05)
DEfound_soft = rownames(PP)[which(PP[,"PPDE"] >= SoftThre)]
```

DenNHist

Density plot to compare the empirical q’s and the simulated q’s from the fitted beta distribution.

Description

'DenNHist' gives the density plot that compares the empirical q’s and the simulated q’s from the fitted beta distribution.

Usage

```r
DenNHist(EBOut, GeneLevel = F)
```

Arguments

- `EBOut`: The output of EBTTest or EBMultiTest.
- `GeneLevel`: Indicate whether the results are from data at gene level.

Value

For data with n1 conditions and n2 uncertainty groups, n1*n2 plots will be generated. Each plot represents a subset of the data. The empirical estimation of q’s will be represented as blue histograms and the density of the fitted beta distribution will be represented as the green line.

Author(s)

Ning Leng

References


See Also

beta.mom, QQP, EBTTest, EBMultiTest
**Examples**

```r
data(GeneMat)
GeneMat.small = GeneMat[c(500:1000),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small,
Conditions = as.factor(rep(c("C1","C2"), each=5)),
sizeFactors = Sizes, maxround = 5)
par(mfrow = c(2,2))
DenNHist(EBOut)
```

**EBMultiTest**  
*Using EM algorithm to calculate the posterior probabilities of interested patterns in a multiple condition study*

**Description**

`EBMultiTest` is built based on the assumption of NB-Beta Empirical Bayes model. It utilizes the EM algorithm to give the posterior probability of the interested patterns.

**Usage**

```r
EBMultiTest(Data, NgVector = NULL, Conditions, AllParti = NULL,
sizeFactors, maxround, Pool = F, NumBin = 1000,
ApproxVal=10^-10, PoolLower=.25, PoolUpper = .75, Print=T,Qtrm=1,QtrmCut=0)
```

**Arguments**

- **Data**  
  A data matrix contains expression values for each transcript (gene or isoform level). In which rows should be transcripts and columns should be samples.

- **NgVector**  
  A vector indicates the uncertainty group assignment of each isoform. e.g. if we use number of isoforms in the host gene to define the uncertainty groups, suppose the isoform is in a gene with 2 isoforms, Ng of this isoform should be 2. The length of this vector should be the same as the number of rows in Data. If it’s gene level data, Ngvector could be left as NULL.

- **Conditions**  
  A vector indicates the condition in which each sample belongs to.

- **AllParti**  
  A matrix indicates the interested patterns. Columns shoule be conditions and rows should be patterns. The matrix could be obtained by the GetPatterns function. If AllParti=NULL, all possible patterns will be used.

- **sizeFactors**  
  The normalization factors. It should be a vector with lane specific numbers (the length of the vector should be the same as the number of samples, with the same order as the columns of Data).

- **maxround**  
  Number of iterations. The default value is 5. Users should always check the convergency by looking at the Alpha and Beta in output. If the hyper-parameter estimations are not converged in 5 iterations, larger number is suggested.

- **Pool**  
  While working without replicates, user could define the Pool = TRUE in the EBTest function to enable pooling.

- **NumBin**  
  By defining NumBin = 1000, EBSeq will group the genes with similar means together into 1,000 bins.
With the assumption that only subset of the genes are DE in the data set, we take genes whose FC are in the PoolLower - PoolUpper quantile of the FC’s as the candidate genes (default is 25%-75%).

For each bin, the bin-wise variance estimation is defined as the median of the cross condition variance estimations of the candidate genes within that bin.

We use the cross condition variance estimations for the candidate genes and the bin-wise variance estimations of the host bin for the non-candidate genes.

ApproxVal

The variances of the transcripts with mean < var will be approximated as mean/(1 - ApproxVal).

Print

Whether print the elapsed-time while running the test.

Qtrm, QtrmCut

Transcripts with Qtrm th quantile <= QtrmCut will be removed before testing. The default value is Qtrm = 1 and QtrmCut=0. By default setting, transcripts with all 0’s won’t be tested.

**Value**

Alpha

Fitted parameter alpha of the prior beta distribution. Rows are the values for each iteration.

Beta

Fitted parameter beta of the prior beta distribution. Rows are the values for each iteration.

P, PFromZ

The bayes estimator of following each pattern of interest. Rows are the values for each iteration.

Z, PoissonZ

The Posterior Probability of following each pattern of interest for each transcript. (Maybe not in the same order of input).

RList

The fitted values of r for each transcript.

MeanList

The mean of each transcript. (across conditions).

VarList

The variance of each transcript. (across conditions).

QList

The fitted q values of each transcript within each condition.

SPMean

The mean of each transcript within each condition (adjusted by the normalization factors).

SPEstVar

The estimated variance of each transcript within each condition (adjusted by the normalization factors).

PoolVar

The variance of each transcript (The pooled value of within condition EstVar).

DataList

A List of data that grouped with Ng and bias.

PPpattern

The Posterior Probability of following each pattern (columns) for each transcript (rows). Transcripts with expression 0 for all samples are not shown in this matrix.

f

The likelihood of likelihood of prior predictive distribution of being each pattern for each transcript.

AllParti

The matrix describe the patterns.

PPpatternWith0

The Posterior Probability of following each pattern (columns) for each transcript (rows). Transcripts with expression 0 for all samples are shown in this matrix with PP(any_pattn)=NA.

ConditionOrder

The condition assignment for C1Mean, C2Mean, etc.
EBTest

Author(s)
Ning Leng

References

See Also
EBTest, GetMultiPP, GetMultiFC

Examples

```r
data(MultiGeneMat)
MultiGeneMat.small = MultiGeneMat[201:210,]
Conditions = c("C1","C1","C2","C2","C3","C3")
PosParti = GetPatterns(Conditions)
Parti = PosParti[3,]
MultiSize = MedianNorm(MultiGeneMat.small)
MultiOut = EBMultiTest(MultiGeneMat.small, NgVector = NULL,

EBTest(Data, NgVector = NULL, Conditions, sizeFactors, maxround,
Pool = F, NumBin = 1000, ApproxVal = 10^-10, Alpha = NULL,
Beta = NULL, PInput = NULL, RInput = NULL,
PoolLower = .25, PoolUpper = .75, Print = T, Qtrm = 1, QtrmCut=0)
```

EBTest

Using EM algorithm to calculate the posterior probabilities of being DE

Description
Base on the assumption of NB-Beta Empirical Bayes model, the EM algorithm is used to get the posterior probability of being DE.

Usage

```r
EBTest(Data, NgVector = NULL, Conditions, sizeFactors, maxround,
Pool = F, NumBin = 1000, ApproxVal = 10^-10, Alpha = NULL,
Beta = NULL, PInput = NULL, RInput = NULL,
PoolLower = .25, PoolUpper = .75, Print = T, Qtrm = 1, QtrmCut=0)
```

Arguments

- **Data**: A data matrix contains expression values for each transcript (gene or isoform level). In which rows should be transcripts and columns should be samples.
- **NgVector**: A vector indicates the uncertainty group assignment of each isoform. e.g. if we use number of isoforms in the host gene to define the uncertainty groups, suppose the isoform is in a gene with 2 isoforms, Ng of this isoform should be 2. The length of this vector should be the same as the number of rows in Data. If it’s gene level data, Ngvector could be left as NULL.
- **Conditions**: A factor indicates the condition which each sample belongs to.
sizeFactors  The normalization factors. It should be a vector with lane specific numbers (the
length of the vector should be the same as the number of samples, with the same
order as the columns of Data).

maxround  Number of iterations. The default value is 5. Users should always check the
convergency by looking at the Alpha and Beta in output. If the hyper-parameter
estimations are not converged in 5 iterations, larger number is suggested.

Pool  While working without replicates, user could define the Pool = TRUE in the
EBTest function to enable pooling.

NumBin  By defining NumBin = 1000, EBSeq will group the genes with similar means
together into 1,000 bins.

PoolLower, PoolUpper  With the assumption that only subset of the genes are DE in the data set, we take
genes whose FC are in the PoolLower - PoolUpper quantile of the FC’s as the
candidate genes (default is 25%-75%).
For each bin, the bin-wise variance estimation is defined as the median of the
cross condition variance estimations of the candidate genes within that bin.
We use the cross condition variance estimations for the candidate genes and the
bin-wise variance estimations of the host bin for the non-candidate genes.

ApproxVal  The variances of the transcripts with mean < var will be approximated as mean/(1-
ApproxVal).

Alpha, Beta, PInput, RInput  If the parameters are known and the user doesn’t want to estimate them from the
data, user could specify them here.

Print  Whether print the elapsed-time while running the test.

Qtrm, QtrmCut  Transcripts with Qtrm th quantile = QtrmCut will be removed before testing.
The default value is Qtrm = 1 and QtrmCut=0. By default setting, transcripts
with all 0’s won’t be tested.

Details
For each transcript gi within condition, the model assumes: X_gistmu_gi ~ NB (r_gi0 * l_s, q_gi)
q_gialpha, beta^N_g ~ Beta (alpha, beta^N_g) In which the l_s is the sizeFactors of samples.
The function will test "H0: q_gi^C1 = q_gi^C2" and "H1: q_gi^C1 != q_gi^C2."

Value
Alpha  Fitted parameter alpha of the prior beta distribution. Rows are the values for
each iteration.

Beta  Fitted parameter beta of the prior beta distribution. Rows are the values for each
iteration.

P, PFromZ  The bayes estimator of being DE. Rows are the values for each iteration.

Z, PoissonZ  The Posterior Probability of being DE for each transcript(Maybe not in the same
order of input).

RList  The fitted values of r for each transcript.

MeanList  The mean of each transcript (across conditions).

VarList  The variance of each transcript (across conditions).

QListi1  The fitted q values of each transcript within condition 1.

QListi2  The fitted q values of each transcript within condition 2.
C1Mean  The mean of each transcript within Condition 1 (adjusted by normalization factors).
C2Mean  The mean of each transcript within Condition 2 (adjusted by normalization factors).
C1EstVar The estimated variance of each transcript within Condition 1 (adjusted by normalization factors).
C2EstVar The estimated variance of each transcript within Condition 2 (adjusted by normalization factors).
PoolVar  The variance of each transcript (The pooled value of within condition EstVar).
DataList A List of data that grouped with Ng.
PPDE    The Posterior Probability of being DE for each transcript (The same order of input).
f0,f1   The likelihood of the prior predictive distribution of being EE or DE (in log scale).
AllZeroIndex The transcript with expression 0 for all samples (which are not tested).
PPMat   A matrix contains posterior probabilities of being EE (the first column) or DE (the second column). Rows are transcripts. Transcripts with expression 0 for all samples are not shown in this matrix.
PPMatWith0 A matrix contains posterior probabilities of being EE (the first column) or DE (the second column). Rows are transcripts. Transcripts with expression 0 for all samples are shown as PP(EE) = PP(DE) = NA in this matrix. The transcript order is exactly the same as the order of the input data.
ConditionOrder The condition assignment for C1Mean, C2Mean, etc.
Conditions The input conditions.
DataNorm Normalized expression matrix.

Author(s)
Ning Leng

References

See Also
EBMultiTest, PostFC, GetPPMat

Examples
data(GeneMat)
str(GeneMat)
GeneMat.small = GeneMat[c(1:10,511:550),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small, Conditions = as.factor(rep(c("C1","C2"), each = 5)), sizeFactors = Sizes, maxround = 5)
PP = GetPPMat(EBOut)
Description

'f0' gives the Prior Predictive Distribution of being EE.

Usage

f0(Input, AlphaIn, BetaIn, EmpiricalR, NumOfGroups, log)

Arguments

Input Expression Values.
AlphaIn, BetaIn, EmpiricalR The parameters estimated from last iteration of EM.
NumOfGroups How many transcripts within each Ng group.
log If true, will give the log of the output.

Value

The function will return the prior predictive distribution values of being EE.

Author(s)

Ning Leng

References


See Also

f1

Examples

#
#f0(matrix(rnorm(100,100,1),ncol=10), .5, .6,
# matrix(rnorm(100,200,1),ncol=10), 100, TRUE)
The Prior Predictive Distribution of being DE

Description

‘f1’ gives the Prior Predictive Distribution of DE.

Usage

f1(Input1, Input2, AlphaIn, BetaIn, EmpiricalRSP1, EmpiricalRSP2, NumOfGroup, log)

Arguments

Input1      Expressions from Condition1.
Input2      Expressions from Condition2.
AlphaIn, BetaIn, EmpiricalRSP1, EmpiricalRSP2
            The parameters estimated from last iteration of EM.
NumOfGroup  How many transcripts within each Ng group.
log         If true, will give the log of the output.

Value

The function will return the prior predictive distribution values of being DE.

Author(s)

Ning Leng

References


See Also

f0

Examples

#f1(matrix(rnorm(100,100,1),ncol=10),
# matrix(rnorm(100,100,1),ncol=10), .5, .6,
# matrix(rnorm(100,200,1),ncol=10),
# matrix(rnorm(100,200,1),ncol=10), 100, TRUE)
The simulated data for two condition gene DE analysis

Description

'GeneMat' gives the simulated data for two condition gene DE analysis.

Usage

data(GeneMat)

Source


See Also

IsoList

Examples

data(GeneMat)

GetDEResults Obtain Differential Expression Analysis Results in a Two-condition Test

Description

Obtain DE analysis results in a two-condition test using the output of EBTtest()

Usage

GetDEResults(EBPrelim, FDR=0.05, Method="robust", FDRMethod="hard", Threshold_FC=0.7, Threshold_FCRatio=0.3, SmallNum=0.01)

Arguments

EBPrelim Output from the function EBTtest().
FDR Target FDR, default is 0.05.
FDRMethod "hard" or "soft". Giving a target FDR alpha, either hard threshold and soft threshold may be used. If the hard threshold is preferred, DE transcripts are defined as the the transcripts with PP(DE) greater than (1-alpha). Using the hard threshold, any DE transcript in the list has FDR <= alpha. If the soft threshold is preferred, the DE transcripts are defined as the transcripts with PP(DE) greater than crit_fun(PPEE, alpha). Using the soft threshold, the list of DE transcripts has average FDR alpha.
GetDEResults

Based on results from our simulation studies, hard thresholds provide a better-controlled empirical FDR when sample size is relatively small (less than 10 samples in each condition). User may consider the soft threshold when sample size is large to improve power.

**Method**

"robust" or "classic". Using the "robust" option, EBSeq is more robust to genes with outliers and genes with extremely small variances. Using the "classic" option, the results will be more comparable to those obtained by using the GetPPMat() function from earlier version (<= 1.7.0) of EBSeq. Default is "robust".

**Threshold_FC**

Threshold for the fold change (FC) statistics. The default is 0.7. The FC statistics are calculated as follows. First the posterior FC estimates are calculated using PostFC() function. The FC statistics is defined as exp(-|log posterior FC|) and therefore is always less than or equal to 1. The default threshold was selected as the optimal threshold learned from our simulation studies. By setting the threshold as 0.7, the expected FC for a DE transcript is less than 0.7 (or greater than 1/0.7=1.4). User may specify their own threshold here. A higher (less conservative) threshold may be used here when sample size is large. Our simulation results indicated that when there are more than or equal to 5 samples in each condition, a less conservative threshold will improve the power when the FDR is still well-controlled. The parameter will be ignored if Method is set as "classic".

**Threshold_FCRatio**

Threshold for the fold change ratio (FCRatio) statistics. The default is 0.3. The FCRatio statistics are calculated as follows. First we get another revised fold change statistic called Median-FC statistic for each transcript. For each transcript, we calculate the median of normalized expression values within each condition. The MedianFC is defined as exp(-|log((C1Median+SmallNum)/(C2Median+SmallNum)))). Note a small number is added to avoid Inf and NA. See SmallNum for more details. The FCRatio is calculated as exp(-|log(FCstatistics/MedianFC)|). Therefore it is always less than or equal to 1. The default threshold was selected as the optimal threshold learned from our simulation studies. By setting the threshold as 0.3, the FCRatio for a DE transcript is expected to be larger than 0.3.

**SmallNum**

When calculating the FCRatio (or Median-FC), a small number is added for each transcript in each condition to avoid Inf and NA. Default is 0.01.

**Details**

GetDEResults() function takes output from EBTest() function and output a list of DE transcripts under a target FDR. It also provides posterior probability estimates for each transcript.

**Value**

<table>
<thead>
<tr>
<th>DEfound</th>
<th>A list of DE transcripts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPMat</td>
<td>Posterior probability matrix. Transcripts are following the same order as in the input matrix. Transcripts that were filtered by magnitude (in EBTest function), FC, or FCR are assigned with NA for both PPDE and PPEE.</td>
</tr>
<tr>
<td>Status</td>
<td>Each transcript will be assigned with one of the following values: &quot;DE&quot;, &quot;EE&quot;, &quot;Filtered: Low Expression&quot;, &quot;Filtered: Fold Change&quot; and &quot;Filtered: Fold Change Ratio&quot;. Transcripts are following the same order as in the input matrix.</td>
</tr>
</tbody>
</table>

**Author(s)**

Ning Leng, Yuan Li
GetMultiFC

Calculate the Fold Changes for Multiple Conditions

Description

`GetMultiFC` calculates the Fold Changes for each pair of conditions in a multiple condition study.

Usage

```r
GetMultiFC(EBMultiOut, SmallNum = 0.01)
```

Arguments

- `EBMultiOut`: The output of `EBMultiTest` function.
- `SmallNum`: A small number will be added for each transcript in each condition to avoid Inf and NA. Default is 0.01.

Details

Provide the FC (adjusted by the normalization factors) for each pair of comparisons. A small number will be added for each transcript in each condition to avoid Inf and NA. Default is set to be 0.01.

Value

- `FCMat`: The FC of each pair of comparison (adjusted by the normalization factors).
- `Log2FCMat`: The log 2 FC of each pair of comparison (adjusted by the normalization factors).
- `PostFCMat`: The posterior FC of each pair of comparison.
- `Log2PostFCMat`: The log 2 posterior FC of each pair of comparison.
- `CondMean`: The mean of each transcript within each condition (adjusted by the normalization factors).
- `ConditionOrder`: The condition assignment for C1Mean, C2Mean, etc.
Author(s)  
Ning Leng

References  

See Also  
EBMultiTest, PostFC

Examples  
data(MultiGeneMat)
MultiGeneMat.small = MultiGeneMat[201:210,]

Conditions = c("C1","C1","C2","C2","C3","C3")

PosParti = GetPatterns(Conditions)
Parti = PosParti[-3,]

MultiSize = MedianNorm(MultiGeneMat.small)

MultiOut = EBMultiTest(MultiGeneMat.small, NgVector=NULL, Conditions=Conditions, AllParti=Parti, sizeFactors=MultiSize, maxround=5)

MultiFC = GetMultiFC(MultiOut)

GetMultiPP  
Posterior Probability of Each Transcript

Description  
'GetMultiPP' generates the Posterior Probability of being each pattern of each transcript based on the EBMultiTest output.

Usage  
GetMultiPP(EBout)

Arguments  
EBout  
The output of EBMultiTest function.

Value  
PP  
The poster probabilities of being each pattern.
MAP  
Gives the most likely pattern.
Patterns  
The Patterns.
GetNg

Author(s)
Ning Leng

References

See Also
GetPPMat

Examples

data(MultiGeneMat)
MultiGeneMat.small = MultiGeneMat[201:210,]

Conditions = c("C1","C1","C2","C2","C3","C3")
PosParti = GetPatterns(Conditions)
Parti = PosParti[-3,]
MultiSize = MedianNorm(MultiGeneMat.small)

MultiOut = EBMultiTest(MultiGeneMat.small, NgVector=NULL, Conditions=Conditions, AllParti=Parti, sizeFactors=MultiSize, maxround=5)
MultiPP = GetMultiPP(MultiOut)

<table>
<thead>
<tr>
<th>GetNg</th>
<th>Ng Vector</th>
</tr>
</thead>
</table>

Description

`GetNg` generates the Ng vector for the isoform level data. (While using the number of isoform in the host gene to define the uncertainty groups.)

Usage

GetNg(IsoformName, GeneName, TrunThre = 3)

Arguments

| IsoformName | A vector contains the isoform names. |
| GeneName    | The gene names of the isoforms in IsoformNames (Should be in the same order). |
| TrunThre    | The number of uncertainty groups the user wish to define. The default is 3. |
Value

GeneNg
The number of isoforms that are contained in each gene.

GeneNgTrun
The truncated Ng of each gene. (The genes contain more than 3 isoforms are with Ng 3.)

IsoformNg
The Ng of each isoform.

IsoformNgTrun
The truncated Ng of each isoform (could be used to define the uncertainty group assignment).

Author(s)

Ning Leng

References


Examples

data(IsoList)
IsoMat = IsoList$IsoMat
IsoNames = IsoList$IsoNames
IsosGeneNames = IsoList$IsosGeneNames
IsoSizes = MedianNorm(IsoMat)
NgList = GetNg(IsoNames, IsosGeneNames)

#IsoNgTrun = NgList$IsoformNgTrun
#IsoEBOut = EBTest(Data = IsoMat, NgVector = IsoNgTrun,
# Conditions = as.factor(rep(c("C1","C2"), each=5)),
# sizeFactors = IsoSizes, maxround = 5)

GetNormalizedMat(Data, Sizes)

Description

‘GetNormalizedMat’ calculates the normalized expression matrix. (Note: this matrix is only used for visualization etc. EBTes and EBMultiTest request *un-adjusted* expressions and normalization factors.)

Usage

GetNormalizedMat(Data, Sizes)

Arguments

Data The data matrix with transcripts in rows and lanes in columns.
Sizes A vector contains the normalization factor for each lane.
GetPatterns

Value

The function will return a normalized matrix.

Author(s)

Ning Leng

References


Examples

data(GeneMat)
str(GeneMat)
Sizes = MedianNorm(GeneMat)
NormData = GetNormalizedMat(GeneMat, Sizes)

---

GetPatterns Generate all possible patterns in a multiple condition study

Description

'GetPatterns' generates all possible patterns in a multiple condition study.

Usage

GetPatterns(Conditions)

Arguments

Conditions The names of the Conditions in the study.

Value

A matrix describe all possible patterns.

Author(s)

Ning Leng

References


Examples

Conditions = c("C1","C1","C2","C2","C3","C3")
PosParti = GetPatterns(Conditions)
GetPP  
Generate the Posterior Probability of each transcript.

Description

'GetPP' generates the Posterior Probability of being DE of each transcript based on the EBTest output.

Usage

GetPP(EBout)

Arguments

EBout  The output of EBTest function.

Value

The poster probabilities of being DE.

Author(s)

Ning Leng

References


See Also

GetPPMat

Examples

data(GeneMat)
GeneMat.small = GeneMat[c(1:10,500:550),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small, Conditions = as.factor(rep(c("C1","C2"), each=5)), sizeFactors = Sizes, maxround = 5)
PPDE = GetPP(EBOut)
str(PPDE)
head(PPDE)
GetPPMat

Description

'GetPPMat' generates the Posterior Probability of being each pattern of each transcript based on the EBTest output.

Usage

GetPPMat(EBout)

Arguments

EBout The output of EBTest function.

Value

The poster probabilities of being EE (first column) and DE (second column).

Author(s)

Ning Leng

References


Examples

data(GeneMat)
GeneMat.small = GeneMat[c(500:550),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small, Conditions = as.factor(rep(c("C1","C2"), each=5)), sizeFactors = Sizes, maxround = 5)
PP = GetPPMat(EBOut)
str(PP)
head(PP)
IsoList

**Description**

'IsoList' gives the simulated data for two condition isoform DE analysis.

**Usage**

```r
data(IsoList)
```

**Source**


**See Also**

GeteMat

**Examples**

```r
data(IsoList)
```

---

IsoMultiList

**Description**

'IsoMultiList' gives a set of simulated data for multiple condition isoform DE analysis.

**Usage**

```r
data(IsoMultiList)
```

**Source**


**See Also**

IsoList

**Examples**

```r
data(IsoMultiList)
```
Likefun

Description

'Likefun' specifies the Likelihood Function of the NB-Beta Model.

Usage

Likefun(ParamPool, InputPool)

Arguments

ParamPool The parameters that will be estimated in EM.
InputPool The control parameters that will not be estimated in EM.

Value

The function will return the log-likelihood.

Author(s)

Ning Leng

References


Examples

```R
#x1 = c(0.6, 0.7, 0.3)
#Input = matrix(rnorm(100, 100, 1), ncol=10)
#RIn = matrix(rnorm(100, 200, 1), ncol=10)
#InputPool = list(Input[,1:5], Input[,6:10], Input,
#rep(.1,100), 1, RIn, RIn[,1:5], RIn[,6:10], 100)
#Likefun(x1, InputPool)
```

LikefunMulti

Description

'LikefunMulti' specifies the Likelihood Function of the NB-Beta Model In Multiple Condition Test.

Usage

LikefunMulti(ParamPool, InputPool)
Arguments

ParamPool  The parameters that will be estimated in EM.
InputPool  The control parameters that will not be estimated in EM.

Value

The function will return the log-likelihood.

Author(s)

Ning Leng

References


Examples

```r
#x1 = c(.6,.7,.3)
#Input = matrix(rnorm(100,100,1),ncol=10)
#RIn = matrix(rnorm(100,200,1),ncol=10)
#InputPool = list(list(Input[,1:5],Input[,6:10]),
# Input, cbind(rep(.1, 10), rep(.9,10)), 1,
# RIn, list(RIn[,1:5],RIn[,6:10]),
# 10, rbind(c(1,1),c(1,2)))
#LikefunMulti(x1, InputPool)
```

---

LogN  

The function to run EM (one round) algorithm for the NB-beta model.

Description

`LogN` specifies the function to run (one round of) the EM algorithm for the NB-beta model.

Usage

LogN(Input, InputSP, EmpiricalR, EmpiricalRSP, NumOfEachGroup, AlphaIn, BetaIn, PIn, NoneZeroLength)

Arguments

Input, InputSP  The expressions among all the samples.
NumOfEachGroup  Number of genes in each Ng group.
AlphaIn, PIn, BetaIn, EmpiricalR, EmpiricalRSP  The parameters from the last EM step.
NoneZeroLength  Number of Ng groups.
LogNMuli

Author(s)

Ning Leng

References


Examples

```r
#Input = matrix(rnorm(100,100,1), ncol=10)
#rownames(Input) = paste("g",1:10)
#RIn = matrix(rnorm(100,200,1), ncol=10)
#res = LogN(Input, list(Input[,1:5], Input[,6:10]),
# RIn, list(RIn[,1:5], RIn[,6:10]),
# 10, .6, .7, .3, 1)
```

---

**LogNMulti**

**EM algorithm for the NB-beta model in the multiple condition test**

Description

'LogNMulti' specifies the function to run (one round of) the EM algorithm for the NB-beta model in the multiple condition test.

Usage

```r
LogNMulti(Input, InputSP, EmpiricalR, EmpiricalRSP, NumOfEachGroup, AlphaIn, BetaIn, PIN,
NoneZeroLength, AllParti, Conditions)
```

Arguments

- **Input, InputSP**: The expressions among all the samples.
- **NumOfEachGroup**: Number of genes in each Ng group.
- **AlphaIn, PIN, BetaIn, EmpiricalR, EmpiricalRSP**: The parameters from the last EM step.
- **NoneZeroLength**: Number of Ng groups.
- **AllParti**: The patterns of interests.
- **Conditions**: The condition assignment for each sample.

Author(s)

Ning Leng
MedianNorm

**References**


**Examples**

```r
#Input = matrix(rnorm(100,100,1),ncol=10)
rownames(Input) = paste("g",1:10)
RIn = matrix(rnorm(100,200,1), ncol=10)
res = LogNMulti(Input, list(Input[,1:5], Input[,6:10]),
# RIn, list(RIn[,1:5], RIn[,6:10]), 10, .6, .7,
# c(.3,.7), 1, rbind(c(1,1), c(1,2)),
# as.factor(rep(c("C1","C2"), each=5))
```

**Description**

`MedianNorm` specifies the median-by-ratio normalization function from Anders et al., 2010.

**Usage**

```
MedianNorm(Data, alternative = FALSE)
```

**Arguments**

- **Data**: The data matrix with transcripts in rows and lanes in columns.
- **alternative**: if `alternative = TRUE`, the alternative version of median normalization will be applied. The alternative method is similar to median-by-ratio normalization, but can deal with the cases when all of the genes/isoforms have at least one zero counts (in which case the median-by-ratio normalization will fail). In more details, in median-by-ratio normalization (denote \( l_1 \) as libsize for sample 1 as an example, assume total \( S \) samples):

  \[
  \hat{l}_1 = \text{median}_g [ \frac{X_{g1}}{(X_{g1}X_{g2}...X_{gS})^{-S}} ] \quad (1)
  \]

  which estimates \( l_1 / (l_1 * l_2 * ... * l_S)^{-S} \). Since we have the constrain that \((l_1 * l_2 * ... * l_S) = 1\), equation (1) estimates \( l_1 \). Note (1) could also be written as:

  \[
  \hat{l}_1 = \text{median}_g [ (X_{g1}X_{g1} * X_{g1}/X_{g2} * ...) / X_{g1}/X_{gS})^{-S}]\]

  In the alternative method, we estimate \( l_1/l_{/1}, l_{/1/l_2}, ... l_{/1/l_S} \) individually by taking median\(_g\)(X\(_g1\)/X\(_g1\)), median\(_g\)(X\(_g1\)/X\(_g2\)) ... Then estimate \( l_1 = l_1 / (l_1 * l_2 * ... * l_S)^{-S} \) by taking the geomean of these estimates:

  \[
  \hat{l}_1 = [ \text{median}_g(X_{g1}/X_{g1}) * \text{median}_g(X_{g1}/X_{g2}) * \text{median}_g(X_{g1}/X_{g3}) * ... * \text{median}_g(X_{g1}/X_{gS}) ]^{-S}
  \]

**Value**

The function will return a vector contains the normalization factor for each lane.
MultiGeneMat

Author(s)
Ning Leng

References

See Also
QuantileNorm

Examples

data(GeneMat)
Sizes = MedianNorm(GeneMat)
#EBOut = EBTest(Data = GeneMat,
  # Conditions = as.factor(rep(c("C1","C2"), each=5)),
  # sizeFactors = Sizes, maxround = 5)

MultiGeneMat The simulated data for multiple condition gene DE analysis

Description
'MultiGeneMat' generates a set of the simulated data for multiple condition gene DE analysis.

Usage
data(MultiGeneMat)

Source

See Also
GeneMat

Examples

data(MultiGeneMat)
PlotPattern

**Visualize the patterns**

**Description**

'PlotPattern' generates the visualized patterns before the multiple condition test.

**Usage**

PlotPattern(Patterns)

**Arguments**

Patterns

The output of GetPatterns function.

**Value**

A heatmap to visualize the patterns of interest.

**Author(s)**

Ning Leng

**References**


**Examples**

Conditions = c("C1","C1","C2","C2","C3","C3")
Patterns = GetPatterns(Conditions)
PlotPattern(Patterns)

PlotPostVsRawFC

**Plot Posterior FC vs FC**

**Description**

'PlotPostVsRawFC' helps the users visualize the posterior FC vs FC in a two condition study.

**Usage**

PlotPostVsRawFC(EBOut, FCOut)

**Arguments**

EBOut

The output of EBMultiTest function.

FCOut

The output of PostFC function.
Fit the mean-var relationship using polynomial regression

PolyFitPlot

Value
A figure shows fold change vs posterior fold change.

Author(s)
Ning Leng

References

See Also
PostFC

Examples
```r
data(GeneMat)
GeneMat.small = GeneMat[500:600,]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small,
Conditions = as.factor(rep(c("C1","C2"), each=5)),
sizeFactors = Sizes, maxround = 5)
FC = PostFC(EBOut)
PlotPostVsRawFC(EBOut,FC)
```

PolyFitPlot

Fit the mean-var relationship using polynomial regression

Description
'PolyFitPlot' fits the mean-var relationship using polynomial regression.

Usage
```
PolyFitPlot(X, Y, nterms, xname = "Estimated Mean",
yname = "Estimated Var", pdfname = "",
xlim = c(-1,5), ylim = c(-1,7), ChangeXY = F,
col = "red")
```

Arguments

- **X**
  The first group of values want to be fitted by the polynomial regression (e.g. Mean of the data).

- **Y**
  The second group of values want to be fitted by the polynomial regression (e.g. variance of the data). The length of Y should be the same as the length of X.

- **nterms**
  How many polynomial terms want to be used.

- **xname**
  Name of the x axis.
30

PolyFitPlot

yn name Name of the y axis.

pdfname Name of the plot.

xlim The x limits of the plot.

ylim The y limits of the plot.

ChangeXY If ChangeXY is setted to be TRUE, X will be treated as the dependent variable and Y will be treated as the independent one. Default is FALSE.

col Color of the fitted line.

Value

The PolyFitPlot function provides a smooth scatter plot of two variables and their best fitting line of polynomial regression.

Author(s)

Ning Leng

References


Examples

data(IsoList)
str(IsoList)
IsoMat = IsoList$IsoMat
IsoNames = IsoList$IsoNames
IsosGeneNames = IsoList$IsosGeneNames
IsoSizes = MedianNorm(IsoMat)
NgList = GetNg(IsoNames, IsosGeneNames)

IsoNgTrun = NgList$IsoformNgTrun
#IsoEBOut = EBTest(Data = IsoMat.small,
# NgVector = IsoNgTrun,
# Conditions = as.factor(rep(c("C1","C2"), each=5)),
# sizeFactors = IsoSizes, maxround = 5)

#par(mfrow=c(2,2))
#PolyFitValue = vector("list",3)

#for(i in 1:3)
# PolyFitValue[[i]] = PolyFitPlot(IsoEBOut$C1Mean[[i]],
# isoEBOut$C1EstVar[[i]], 5)

#PolyAll = PolyFitPlot(unlist(IsoEBOut$C1Mean),
# unlist(IsoEBOut$C1EstVar), 5)

#lines(log10(IsoEBOut$C1Mean[[1]][PolyFitValue[[1]]$sort]),
# PolyFitValue[[1]]$fit[PolyFitValue[[1]]$sort],
# col="yellow", lwd=2)
#lines(log10(IsoEBOut$C1Mean[[2]][PolyFitValue[[2]]$sort]),
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],
# PolyFitValue[[2]]$fit[PolyFitValue[[2]]$
### PostFC

*Calculate the posterior fold change for each transcript across conditions.*

Description

‘PostFC’ calculates the posterior fold change for each transcript across conditions.

Usage

```r
PostFC(EBoutput, SmallNum = 0.01)
```

Arguments

- **EBoutput**: The output from function EBTest.
- **SmallNum**: A small number will be added for each transcript in each condition to avoid Inf and NA. Default is 0.01.

Value

Provide both FC and posterior FC across two conditions. FC is calculated as (MeanC1+SmallNum)/(MeanC2+SmallNum). And Posterior FC is calculated as:

```r
# Post alpha P_a_C1 = alpha + r_C1 * n_C1
# Post beta P_b_C1 = beta + Mean_C1 * n_C1
# P_q_C1 = P_a_C1 / (P_a_C1 + P_b_C1)
# Post FC = ((1-P_q_C1)/P_q_c1) / ((1-P_q_c2)/P_q_c2)
```

- **PostFC**: The posterior FC across two conditions.
- **RealFC**: The FC across two conditions (adjusted by the normalization factors).
- **Direction**: The direction of FC calculation.

Author(s)

Ning Leng

References

See Also

EBTest, GetMultiFC

Examples

data(GeneMat)
GeneMat.small = GeneMat[c(500:550),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small,
Conditions = as.factor(rep(c("C1","C2"), each=5)),
sizeFactors = Sizes, maxround = 5)
FC=PostFC(EBOut)

QQP

The Quantile-Quantile Plot to compare the empirical q’s and simulated q’s from fitted beta distribution

Description

’QQP’ gives the Quantile-Quantile Plot to compare the empirical q’s and simulated q’s from fitted beta distribution.

Usage

QQP(EBOut, GeneLevel = F)

Arguments

EBOut The output of EBTest or EBMultiTest.
GeneLevel Indicate whether the results are from data at gene level.

Value

For data with n1 conditions and n2 uncertainty groups, n1*n2 plots will be generated. Each plot represents a subset of the data.

Author(s)

Ning Leng

References


See Also

EBTest, EBMultiTest, DenNHist
Examples

data(GeneMat)
GeneMat.small = GeneMat[c(500:1000),]
Sizes = MedianNorm(GeneMat.small)
EBOut = EBTest(Data = GeneMat.small,
Conditions = as.factor(rep(c("C1","C2"), each=5)),
sizeFactors = Sizes, maxround = 5)
par(mfrow=c(2,2))
QQP(EBOut)

QuantileNorm  Quantile Normalization

Description

'QuantileNorm' gives the quantile normalization.

Usage

QuantileNorm(Data, Quantile)

Arguments

Data
The data matrix with transcripts in rows and lanes in columns.

Quantile
The quantile the user wishes to use. Should be a number between 0 and 1.

Details

Use a quantile point to normalize the data.

Value

The function will return a vector contains the normalization factor for each lane.

Author(s)

Ning Leng

References


See Also

MedianNorm
Examples

data(GeneMat)
Sizes = QuantileNorm(GeneMat,.75)
#EBOut = EBTest(Data = GeneMat,
# Conditions = as.factor(rep(c("C1","C2"), each=5)),
# sizeFactors = Sizes, maxround = 5)

Description

‘RankNorm’ gives the rank normalization.

Usage

RankNorm(Data)

Arguments

Data The data matrix with transcripts in rows and lanes in columns.

Value

The function will return a matrix contains the normalization factor for each lane and each transcript.

Author(s)

Ning Leng

See Also

MedianNorm, QuantileNorm

Examples

data(GeneMat)
Sizes = RankNorm(GeneMat)
# Run EBSeq
# EBres = EBTest(Data = GeneData, NgVector = rep(1,10^4),
# Vect5End = rep(1,10^4), Vect3End = rep(1,10^4),
# Conditions = as.factor(rep(c(1,2), each=5)),
# sizeFactors = Sizes, maxround=5)
Index

*Topic DE
  EBMultiTest, 6
  EBTest, 8
  GetDEResults, 13

*Topic FDR
  crit_fun, 4

*Topic Fold Change
  PostFC, 31

*Topic Multiple Condition
  EBMultiTest, 6

*Topic Ng
  GetNg, 17

*Topic Normalization
  GetNormalizedMat, 18
  MedianNorm, 26
  QuantileNorm, 33
  RankNorm, 34

*Topic Posterior Probability
  GetMultiFC, 15
  GetMultiPP, 16
  GetPP, 20
  GetPPMat, 21
  PlotPostVsRawFC, 28

*Topic Q-Q plot
  QQP, 32

*Topic Two condition
  EBTest, 8
  GetDEResults, 13

*Topic beta
  beta.mom, 3
  DenNHist, 5

*Topic datasets
  GeneMat, 13
  IsoList, 22
  IsoMultiList, 22
  MultiGeneMat, 27

*Topic package
  EBSeq_NingLeng-package, 2

*Topic patterns
  PlotPattern, 28

beta.mom, 3

crit_fun, 4

DenNHist, 5
EBMultiTest, 6
EBSeq_NingLeng
  (EBSeq_NingLeng-package), 2
EBSeq_NingLeng-package, 2
EBTest, 8

f0, 11
f1, 12

GeneMat, 13
GetDEResults, 13
GetMultiFC, 15
GetMultiPP, 16
GetNg, 17
GetNormalizedMat, 18
GetPatterns, 19
GetPP, 20
GetPPMat, 21

IsoList, 22
IsoMultiList, 22

Likefun, 23
LikefunMulti, 23
LogN, 24
LogNMult, 25

MedianNorm, 26
MultiGeneMat, 27

PlotPattern, 28
PlotPostVsRawFC, 28
PolyFitPlot, 29
PostFC, 31

QQP, 32
QuantileNorm, 33

RankNorm, 34