

Package ‘metaR’

March 30, 2013

Title Statistical analyses for high-throughput sequencing

Version 0.99.0

Date 2012-07-23

Author Joseph Nathaniel Paulson, Hector Corrada-Bravo

Maintainer Joseph Paulson <jpaulson@umiacs.umd.edu>

Description metaR is designed to determine features (be it Operational Taxonomic Unit (OTU), species, etc.) that are differentially abundant between two or more groups of multiple samples. metaR is designed to address the effects of both normalization and under-sampling of microbial communities on disease association detection and the testing of feature correlations.

License Artistic-2.0

Depends R(>= 2.10.0), Biobase, limma, matrixStats, methods, RColorBrewer, gplots

Suggests annotate

Collate

‘zigControl.R’ ‘cumNorm.R’ ‘plotOTU.R’ ‘fitZig.R’ ‘doCountMStep.R’ ‘doZeroMStep.R’ ‘doEStep.R’ ‘getZ.R’ ‘getPi.R’

R topics documented:

aggregateM	2
cumNorm	3
cumNormMat	3
cumNormStat	4
doCountMStep	5
doEStep	6
doZeroMStep	6
exportMat	7
exportStats	8
expSummary	9
fitZig	9

getCountDensity	10
getEpsilon	11
getNegativeLogLikelihoods	12
getPi	12
getZ	13
isItStillActive	14
libSize	14
load_meta	15
load_metaQ	16
load_phenoData	16
lungData	17
mouseData	17
MRcoefs	18
MRcounts	19
MRexperiment	20
MRfisher	21
MRfulltable	22
MRtable	23
newMRexperiment	24
normFactors	25
plotCorr	26
plotGenus	27
plotMRheatmap	28
plotOTU	28
posterior.probs	29
zigControl	30

Index 31

aggregateM	<i>Aggregates counts by a particular classification.</i>
------------	--

Description

This function takes a MRexperiment object of data at a particular level with feature information allowing for aggregation of counts to a particular level. This method assumes taxa begin at the highest level and continue to the current level, reverse assumes taxa begin at the lowest level.

Usage

```
aggregateM(obj, taxa, lvl, split=";")
```

Arguments

obj	A MRexperiment object.
lvl	The level to go up (numeric, 1,2,3).
taxa	A vector of taxa annotations with splits
split	The way character strings in taxa in the obj are split.

Value

Updated object with counts aggregated to the various taxonomic levels.

cumNorm	<i>Cumulative sum scaling factors.</i>
---------	--

Description

Calculates each column's quantile and calculates the sum up to and including that quantile.

Usage

```
cumNorm(obj, p = cumNormStat(obj))
```

Arguments

obj	An MRexperiment object.
p	The pth quantile.

Value

Vector of the sum up to and including a sample's pth quantile

See Also

[fitZig](#) [cumNormStat](#)

Examples

```
data(mouseData)
cumNorm(mouseData)
head(normFactors(mouseData))
```

cumNormMat	<i>Cumulative sum scaling factors.</i>
------------	--

Description

Calculates each column's quantile and calculates the sum up to and including that quantile.

Usage

```
cumNormMat(obj, p = cumNormStat(obj))
```

Arguments

obj A MRExperiment object.
 p The pth quantile.

Value

Returns a matrix normalized by scaling counts up to and including the pth quantile.

See Also

[fitZig](#) [cumNorm](#)

Examples

```
data(mouseData)
head(cumNormMat(mouseData))
```

cumNormStat	<i>Cumulative sum scaling percentile selection</i>
-------------	--

Description

Calculates the percentile for which to sum counts up to and scale by.

Usage

```
cumNormStat(obj, pFlag = FALSE, rel=.1, qFlag = TRUE, ...)
```

Arguments

obj A list with count data
 pFlag Plot the median difference quantiles
 rel Cutoff for the relative difference from one median difference from the reference to the next
 qFlag Flag to either calculate the proper percentile using a step-wise or triangular approximation of the sample count distribution.
 ... Applicable if pFlag == TRUE. Extra plotting parameters.

Value

Percentile for which to scale data

See Also

[fitZig](#) [cumNorm](#)

Examples

```
data(mouseData)
p = round(cumNormStat(mouseData,pFlag=FALSE),digits=2)
s95=cumNorm(mouseData)
```

doCountMStep

*Compute the Maximization step calculation for features still active.***Description**

Maximization step is solved by weighted least squares. The function also computes counts residuals.

Usage

```
doCountMStep(z, y, mmCount, stillActive, fit2 = NULL)
```

Arguments

<code>z</code>	Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).
<code>y</code>	Matrix (m x n) of count observations.
<code>mmCount</code>	Model matrix for the count distribution.
<code>stillActive</code>	Boolean vector of size M, indicating whether a feature converged or not.
<code>fit2</code>	Previous fit of the count model.

Details

Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij} = \pi_j(S_j) * f_0(y_{ij}) + (1 - \pi_j(S_j)) * f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2))$. The log-likelihood in this extended model is $(1 - \delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1 - \delta_{ij}) \log (1 - \pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij} = 1 | \text{data})$.

Value

Update matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).

See Also

[fitZig](#)

doEStep *Compute the Expectation step.*

Description

Estimates the responsibilities $z_{ij} = \frac{\pi_j \cdot I_0(y_{ij})}{\pi_j \cdot I_0(y_{ij}) + (1 - \pi_j) \cdot f_{\text{count}}(y_{ij})}$

Usage

```
doEStep(countResiduals, zeroResiduals, zeroIndices)
```

Arguments

countResiduals Residuals from the count model.
 zeroResiduals Residuals from the zero model.
 zeroIndices Index (matrix m x n) of counts that are zero/non-zero.

Details

Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij}) = \pi_j \cdot I_0(y_{ij}) + (1 - \pi_j) \cdot f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2)$. The log-likelihood in this extended model is $(1 - \delta_{ij}) \log f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j + (1 - \delta_{ij}) \log (1 - \pi_j)$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij} = 1 \mid \text{data})$.

Value

Updated matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).

See Also

[fitZig](#)

doZeroMStep *Compute the zero Maximization step.*

Description

Performs Maximization step calculation for the mixture components. Uses least squares to fit the parameters of the mean of the logistic distribution. $\pi_j = \sum_i \frac{1}{M} M_{z_{ij}}$ Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij} = \pi_j(S_j) \cdot f_0(y_{ij}) + (1 - \pi_j(S_j)) \cdot f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2)$. The log-likelihood in this extended model is $(1 - \delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1 - \delta_{ij}) \log (1 - \pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij} = 1 | \text{data})$.

Usage

```
doZeroMStep(z, zeroIndices, mmZero)
```

Arguments

z	Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).
zeroIndices	Index (matrix m x n) of counts that are zero/non-zero.
mmZero	The zero model, the model matrix to account for the change in the number of OTUs observed as a linear effect of the depth of coverage.

Value

List of the zero fit (zero mean model) coefficients, variance - scale parameter (scalar), and normalized residuals of length $\text{sum}(\text{zeroIndices})$.

See Also

[fitZig](#)

exportMat	<i>export the normalized eSet dataset as a matrix.</i>
-----------	--

Description

This function allows the user to take a dataset of counts and output the dataset to the user's workspace as a tab-delimited file, etc.

Usage

```
exportMat(mat, output = "~/Desktop/matrix.tsv")
```

Arguments

mat	A matrix of values (normalized, or otherwise)
output	Output file name

Value

NA

See Also[cumNorm](#)**Examples**

```
# Not run
#data(mouseData)
#exportMat(MRcounts(mouseData)[1:5,1:5],output="~/Desktop/normMatrix.tsv");
```

`exportStats`*Various statistics of the count data.*

Description

A matrix of values for each sample. The matrix consists of sample ids, the sample scaling factor, quantile value, and the number of number of features.

Usage

```
exportStats(obj, p = cumNormStat(obj),
  output = "~/Desktop/res.stats.tsv")
```

Arguments

<code>obj</code>	A MRexperiment object with count data.
<code>p</code>	Quantile value to calculate the scaling factor and quantiles for the various samples.
<code>output</code>	Output file name.

Value

None.

See Also[cumNorm quantile](#)**Examples**

```
# Not run
#data(mouseData)
#exportStats(mouseData,p=1,output="~/Desktop/mouseData-stats.tsv")
```

 expSummary

Access MRExperiment object experiment data

Description

The expSummary vectors represent the column (sample specific) sums of features, i.e. the total number of reads for a sample, libSize and also the normalization factors, normFactor.

Usage

```
## S4 method for signature 'MRExperiment'
expSummary(obj)
```

Arguments

obj a MRExperiment object.

Author(s)

Joseph N. Paulson, jpaulson@umiacs.umd.edu

Examples

```
#load(mouseData)
#expSummary(mouseData)
```

 fitZig

Computes the weighted fold-change estimates and t-statistics.

Description

Wrapper to actually run the Expectation-maximization algorithm and estimate f_{count} fits. Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij} = \pi_j(S_j) * f_0(y_{ij}) + (1 - \pi_j(S_j)) * f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2)$. The log-likelihood in this extended model is: $(1 - \delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1 - \delta_{ij}) \log (1 - \pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij} = 1 \mid \text{data})$.

Usage

```
fitZig(obj, mod, zeroMod = NULL,
       useS95offset = TRUE, control = zigControl())
```

Arguments

obj	A MRexperiment object with count data.
mod	The model for the count distribution.
zeroMod	The zero model, the model to account for the change in the number of OTUs observed as a linear effect of the depth of coverage.
useS95offset	Boolean, whether to include the default scaling parameters in the model or not.
control	The settings for fitZig.

Value

The fits, posterior probabilities, posterior probabilities used at time of convergence for each feature, ebayes (limma object) fit, among other data.

See Also

[cumNorm](#) [zigControl](#)

Examples

```
data(lungData)
k = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[,-k]
cumNorm(lungTrim)
k = which(rowSums(MRcounts(lungTrim)>0)<10)
lungTrim = lungTrim[-k,]
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
settings = zigControl(maxit=1, verbose=FALSE)
fit = fitZig(obj = lungTrim, mod=mod, control=settings)
```

getCountDensity	<i>Compute the value of the count density function from the count model residuals.</i>
-----------------	--

Description

Calculate density values from a normal: $f(x) = 1/(\sqrt{2\pi}) \sigma^{-1} e^{-((x - \mu)^2/(2\sigma^2))}$. Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij} = \pi_j(S_j) \cdot f_0(y_{ij}) + (1 - \pi_j(S_j)) \cdot f_{\text{count}}(y_{ij}; \mu_i, \sigma_i^2)$. The log-likelihood in this extended model is $(1 - \delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1 - \delta_{ij}) \log (1 - \pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij} = 1 \mid \text{data})$.

Usage

```
getCountDensity(residuals, log = FALSE)
```

Arguments

residuals	Residuals from the count model.
log	Whether or not we are calculating from a log-normal distribution.

Value

Density values from the count model residuals.

See Also

[fitZig](#)

getEpsilon	<i>Calculate the relative difference between iterations of the negative log-likelihoods.</i>
------------	--

Description

Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The log-likelihood in this extended model is $(1-\delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1-\delta_{ij}) \log (1-\pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij}=1 \mid \text{data})$.

Usage

```
getEpsilon(nll, nllOld)
```

Arguments

nll	Vector of size M with the current negative log-likelihoods.
nllOld	Vector of size M with the previous iterations negative log-likelihoods.

Value

Vector of size M of the relative differences between the previous and current iteration nll.

See Also

[fitZig](#)

```
getNegativeLogLikelihoods
```

Calculate the negative log-likelihoods for the various features given the residuals.

Description

Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $\delta_{ij} = 1$ if y_{ij} is generated from the zero point mass as latent indicator variables. The log-likelihood in this extended model is $(1-\delta_{ij}) \log f_{\text{count}}(y; \mu_i, \sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1-\delta_{ij}) \log (1-\pi_j(s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij}=1 \mid \text{data and current values})$.

Usage

```
getNegativeLogLikelihoods(z, countResiduals,
  zeroResiduals)
```

Arguments

`z` Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).

`countResiduals` Residuals from the count model.

`zeroResiduals` Residuals from the zero model.

Value

Vector of size M of the negative log-likelihoods for the various features.

See Also

[fitZig](#)

```
getPi
```

Calculate the mixture proportions from the zero model / spike mass model residuals.

Description

$F(x) = 1 / (1 + \exp(-(x-m)/s))$ (the CDF of the logistic distribution). Provides the probability that a real-valued random variable X with a given probability distribution will be found at a value less than or equal to x. The output are the mixture proportions for the samples given the residuals from the zero model.

Usage

```
getPi(residuals)
```

Arguments

`residuals` Residuals from the zero model.

Value

Mixture proportions for each sample.

See Also

[fitZig](#)

<code>getZ</code>	<i>Calculate the current Z estimate responsibilities (posterior probabilities)</i>
-------------------	--

Description

Calculate the current Z estimate responsibilities (posterior probabilities)

Usage

```
getZ(z, zUsed, stillActive, nll, nllUSED)
```

Arguments

<code>z</code>	Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).
<code>zUsed</code>	Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0) that are actually used (following convergence).
<code>stillActive</code>	A vector of size M booleans saying if a feature is still active or not.
<code>nll</code>	Vector of size M with the current negative log-likelihoods.
<code>nllUSED</code>	Vector of size M with the converged negative log-likelihoods.

Value

A list of updated `zUsed` and `nllUSED`.

See Also

[fitZig](#)

`isItStillActive` *Function to determine if a feature is still active.*

Description

In the Expectation Maximization routine features posterior probabilities routinely converge based on a tolerance threshold. This function checks whether or not the feature's negative log-likelihood (measure of the fit) has changed or not.

Usage

```
isItStillActive(eps, tol, stillActive, stillActiveNLL,
               nll)
```

Arguments

<code>eps</code>	Vector of size M (features) representing the relative difference between the new nll and old nll.
<code>tol</code>	The threshold tolerance for the difference
<code>stillActive</code>	A vector of size M booleans saying if a feature is still active or not.
<code>stillActiveNLL</code>	A vector of size M recording the negative log-likelihoods of the various features, updated for those still active.
<code>nll</code>	Vector of size M with the current negative log-likelihoods.

Value

None.

See Also

[fitZig](#)

`libSize` *Access sample depth of coverage from MRExperiment object*

Description

The `libSize` vector represents the column (sample specific) sums of features, i.e. the total number of reads for a sample. It is used by [fitZig](#).

Usage

```
## S4 method for signature 'MRExperiment'
libSize(obj)
```

Arguments

obj a MRexperiment object.

Author(s)

Joseph N. Paulson, jpaulson@umiacs.umd.edu

Examples

```
data(lungData)
head(libSize(lungData))
```

load_meta	<i>Load a count dataset associated with a study.</i>
-----------	--

Description

Load a matrix of OTUs in a tab delimited format

Usage

```
load_meta(file, sep="\t")
```

Arguments

file Path and filename of the actual data file.
sep File delimiter.

Value

An object of count data.

See Also

[load_phenoData](#)

Examples

```
# jobj = load_meta("~/Desktop/testFile.tsv")
```

load_metaQ	<i>Load a count dataset associated with a study set up in a Qiime format.</i>
------------	---

Description

Load a matrix of OTUs in Qiime's format

Usage

```
load_metaQ(file)
```

Arguments

file Path and filename of the actual data file.

Value

An object of count data.

See Also

[load_meta](#) [load_phenoData](#)

Examples

```
# jobj = load_metaQ("~/Desktop/testFile.tsv")
```

load_phenoData	<i>Load a clinical/phenotypic dataset associated with a study.</i>
----------------	--

Description

Load a matrix of metadata associated with a study.

Usage

```
load_phenoData(file, tran = FALSE, sep = "\t")
```

Arguments

file Path and filename of the actual clinical file.

tran Boolean. If the covariates are along the columns and samples along the rows, then tran should equal TRUE.

sep The separator for the file.

Value

The metadata as a dataframe.

See Also

[load_meta](#)

Examples

```
#clin = load_phenoData("~/Desktop/testFile.tsv")
```

lungData	<i>OTU abundance matrix of samples from a smoker/non-smoker study</i>
----------	---

Description

This is a list with a matrix of OTU counts,otu names, taxa annotations for each OTU, and phenotypic data. Samples along the columns and OTUs along the rows.

Usage

```
lungData
```

Format

A list of OTU matrix, taxa, otus, and phenotypes

References

<http://www.ncbi.nlm.nih.gov/pubmed/21680950>

mouseData	<i>OTU abundance matrix of mice samples from a diet longitudinal study</i>
-----------	--

Description

This is a list with a matrix of OTU counts, taxa annotations for each OTU, otu names, and vector of phenotypic data. Samples along the columns and OTUs along the rows.

Usage

```
mouseData
```

Format

A list of OTU matrix, taxa, otus, and phenotypes

References

<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2894525/>

MRcoefs

Table of top-ranked microbial marker gene from linear model fit

Description

Extract a table of the top-ranked features from a linear model fit. This function will be updated soon to provide better flexibility similar to limma's topTable.

Usage

```
MRcoefs(obj, by=2, coef=NULL, number=10, taxa=obj$taxa, adjust.method="fdr", group=0, eff=0, output=NULL)
```

Arguments

obj	A list containing the linear model fit produced by lmFit through fitZig.
by	Column number or column name specifying which coefficient or contrast of the linear model is of interest.
coef	Column number(s) or column name(s) specifying which coefficient or contrast of the linear model to display.
number	The number of bacterial features to pick out.
taxa	Taxa list.
adjust.method	Method to adjust p-values by. Default is "FDR". Options include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". See p.adjust for more details.
group	One of three choices, 0,1,2. 0: the sort is ordered by a decreasing absolute value coefficient fit. 1: the sort is ordered by the raw coefficient fit in decreasing order. 2: the sort is ordered by the raw coefficient fit in increasing order.
eff	Restrict samples to have at least eff quantile effective samples.
output	Name of output file, including location, to save the table.

Value

Table of the top-ranked features determined by the linear fit's coefficient.

See Also

[fitZig MRtable](#)

Examples

```
data(lungData)
k = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[,-k]
k = which(rowSums(MRcounts(lungTrim)>0)<10)
lungTrim = lungTrim[-k,]
cumNorm(lungTrim)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
settings = zigControl(maxit=1, verbose=FALSE)
fit = fitZig(obj = lungTrim, mod=mod, control=settings)
head(MRcoefs(fit))
```

MRcounts*Accessor for the counts slot of a MRexperiment object*

Description

The counts slot holds the raw count data representing (along the rows) the number of reads annotated for a particular feature and (along the columns) the sample.

Usage

```
## S4 method for signature 'MRexperiment'
MRcounts(cnts, norm=FALSE)
```

Arguments

cnts a MRexperiment object.

norm logical indicating whether or not to return normalized counts.

Author(s)

Joseph N. Paulson, jpaulson@umiacs.umd.edu

Examples

```
#load(lungDataset)
#head(MRcounts(lungData))
```

MRExperiment	<i>Class "MRExperiment" – a modified eSet object for the data from high-throughput sequencing experiments</i>
--------------	---

Description

This is the main class for metaR.

Objects from the Class

Objects should be created with calls to `newMRExperiment`.

Extends

Class `eSet` (package 'Biobase'), directly. Class `VersionedBiobase` (package 'Biobase'), by class "eSet", distance 2. Class `Versioned` (package 'Biobase'), by class "eSet", distance 3.

Methods

Class-specific methods.

[<sample>, <variable>]: Subset operation, taking two arguments and indexing the sample and variable. Returns an `MRExperiment` object, including relevant metadata. Setting `drop=TRUE` generates an error. Subsetting the data, the experiment summary slot is repopulated and `pData` is repopulated after calling `factor` (removing levels not present).

Note

Note: This is a summary for reference. For an explanation of the actual usage, see the vignette.

`MRExperiments` are the main class in use by metaR. The class extends `eSet` and provides additional slots which are populated during the analysis pipeline.

`MRExperiment` datasets are created with calls to `newMRExperiment`. `MRExperiment` datasets contain raw count matrices (integers) accessible through `counts`. Similarly, normalized count matrices can be accessed (following normalization) through `counts` by calling `norm=TRUE`. Following an analysis, a matrix of posterior probabilities for counts is accessible through `posterior.probs`.

The normalization factors used in analysis can be recovered by `normFactors`, as can the library sizes of samples (depths of coverage), `libSize`.

Similarly to other RNASeq bioconductor packages available, the rows of the matrix correspond to a feature (be it OTU, species, gene, etc.) and each column an experimental sample. Pertinent clinical information and potential confounding factors are stored in the `phenoData` slot (accessed via `pData`).

To populate the various slots in an `MRExperiment` several functions are run. 1) `cumNormStat` calculates the proper percentile to calculate normalization factors. The `cumNormStat` slot is populated. 2) `cumNorm` calculates the actual normalization factors using `p = cumNormStat`.

Other functions will place subsequent matrices (normalized counts (`cumNormMat`), posterior probabilities (`posterior.probs`))

As mentioned above, MRexperiment is derived from the virtual class, eSet and thereby has a phenoData slot which allows for sample annotation. In the phenoData data frame factors are stored. The normalization factors and library size information is stored in a slot called expSummary that is an annotated data frame and is repopulated for subsetted data.

Examples

```
# See vignette
```

MRfisher

Wrapper to run fisher's test on presence/absence of a feature.

Description

This function returns a data frame of p-values, odds ratios, lower and upper confidence limits for every row of a matrix.

Usage

```
MRfisher(obj, cl, mat=FALSE)
```

Arguments

obj	A MRexperiment object with a count matrix, or a simple count matrix.
cl	Group comparison
mat	logical indicating whether obj is a MRexperiment object or matrix. Default is a MRexperiment object.

Value

NA

See Also

[cumNorm fitZig](#)

Examples

```
data(lungData)
k = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[,-k]
lungTrim = lungTrim[-which(rowSums(MRcounts(lungTrim)>0)<20),]
res = MRfisher(lungTrim, pData(lungTrim)$SmokingStatus);
head(res)
```

MRfulltable	<i>Table of top microbial marker gene from linear model fit including sequence information</i>
-------------	--

Description

Extract a table of the top-ranked features from a linear model fit. This function will be updated soon to provide better flexibility similar to limma's topTable. This function differs from link{MRcoefs} in that it provides other information about the presence or absence of features to help ensure significant features called are moderately present.

Usage

```
MRfulltable(obj,by=2,coef=NULL,number=10,taxa=obj$taxa,adjust.method="fdr",group=0,eff=0,output=NULL)
```

Arguments

obj	A list containing the linear model fit produced by lmFit through fitZig.
by	Column number or column name specifying which coefficient or contrast of the linear model is of interest.
coef	Column number(s) or column name(s) specifying which coefficient or contrast of the linear model to display.
number	The number of bacterial features to pick out.
taxa	Taxa list.
adjust.method	Method to adjust p-values by. Default is "FDR". Options include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". See p.adjust for more details.
group	One of three choices, 0,1,2. 0: the sort is ordered by a decreasing absolute value coefficient fit. 1: the sort is ordered by the raw coefficient fit in decreasing order. 2: the sort is ordered by the raw coefficient fit in increasing order.
eff	Restrict samples to have at least eff quantile effective samples.
output	Name of output file, including location, to save the table.

Value

Table of the top-ranked features determined by the linear fit's coefficient.

See Also

[fitZig](#) [MRcoefs](#) [MRtable](#) [MRfisher](#)

Examples

```

data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[,-k]
k = which(rowSums(MRcounts(lungTrim)>0)<10)
lungTrim = lungTrim[-k,]
cumNorm(lungTrim)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
settings = zigControl(maxit=1,verbose=FALSE)
fit = fitZig(obj = lungTrim,mod=mod,control=settings)
head(MRfulltable(fit))

```

MRtable	<i>Table of top microbial marker gene from linear model fit including sequence information</i>
---------	--

Description

Extract a table of the top-ranked features from a linear model fit. This function will be updated soon to provide better flexibility similar to limma's topTable. This function differs from link{MRcoefs} in that it provides other information about the presence or absence of features to help ensure significant features called are moderately present.

Usage

```
MRtable(obj,by=2,coef=NULL,number=10,taxa=obj$taxa,adjust.method="fdr",group=0,output=NULL)
```

Arguments

obj	A list containing the linear model fit produced by lmFit through fitZig.
by	Column number or column name specifying which coefficient or contrast of the linear model is of interest.
coef	Column number(s) or column name(s) specifying which coefficient or contrast of the linear model to display.
number	The number of bacterial features to pick out.
taxa	Taxa list.
adjust.method	Method to adjust p-values by. Default is "FDR". Options include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". See p.adjust for more details.
group	One of three choices, 0,1,2. 0: the sort is ordered by a decreasing absolute value coefficient fit. 1: the sort is ordered by the raw coefficient fit in decreasing order. 2: the sort is ordered by the raw coefficient fit in increasing order.
output	Name of output file, including location, to save the table.

Value

Table of the top-ranked features determined by the linear fit's coefficient.

See Also

[fitZig MRcoefs](#)

Examples

```
data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[,-k]
k = which(rowSums(MRcounts(lungTrim)>0)<10)
lungTrim = lungTrim[-k,]
cumNorm(lungTrim)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
settings = zigControl(maxit=1,verbose=FALSE)
fit = fitZig(obj = lungTrim,mod=mod,control=settings)
head(MRtable(fit))
```

newMRExperiment

Create a MRExperiment object

Description

This function creates a MRExperiment object from a matrix or data frame of count data.

Usage

```
newMRExperiment(counts, phenoData=NULL, featureData=NULL, libSize=NULL, normFactors=NULL)
```

Arguments

counts	A matrix or data frame of count data. The count data is representative of the number of reads annotated for a feature (be it gene, OTU, species, etc). Rows should correspond to features and columns to samples.
phenoData	An AnnotatedDataFrame with pertinent sample information.
featureData	An AnnotatedDataFrame with pertinent feature information.
libSize	libSize, library size, is the total number of reads for a particular sample.
normFactors	normFactors, the normalization factors used in either the model or as scaling factors of sample counts for each particular sample.

Details

See [MRExperiment-class](#) and eSet (from the Biobase package) for the meaning of the various slots.

Value

an object of class MRExperiment

Author(s)

Joseph N Paulson, jpaulson@umiacs.umd.edu

Examples

```
cnts = matrix(abs(rnorm(1000)),nc=10)
obj <- newMRExperiment(cnts)
```

normFactors

Access the normalization factors in a MRExperiment object

Description

Function to access the scaling factors, aka the normalization factors, of samples in a MRExperiment object.

Usage

```
## S4 method for signature 'MRExperiment'
normFactors(obj)
```

Arguments

obj a MRExperiment object.

Author(s)

Joseph N. Paulson, jpaulson@umiacs.umd.edu

Examples

```
data(lungData)
cumNorm(lungData)
head(normFactors(lungData))
```

plotCorr	<i>Basic correlation plot function for normalized or unnormalized counts.</i>
----------	---

Description

This function plots a heatmap of the "n" features with greatest variance across rows.

Usage

```
plotCorr(obj, n, trials, log=TRUE, norm=TRUE, fun=cor, ...)
```

Arguments

obj	A MRexperiment object with count data.
n	The number of features to plot
trials	A vector of clinical information for.
log	Whether or not to log transform the counts.
norm	Whether or not to normalize the counts.
fun	Function to calculate pair-wise relationships. Default is pearson correlation
...	Additional plot arguments.

Value

NA

See Also

[cumNormMat](#)

Examples

```
data(mouseData)
trials = pData(mouseData)$diet
plotCorr(obj=mouseData, n=200, trials=trials, cexRow = 0.4, cexCol = 0.4, trace="none", dendrogram="none")
```

plotGenus *Basic plot function of the raw or normalized data.*

Description

This function plots the abundance of a particular OTU by class. The function uses the estimated posterior probabilities to make technical zeros transparent.

Usage

```
plotGenus(obj, otuIndex, classIndex, norm = TRUE, no=1:length(otuIndex), jitter = TRUE, factor = 1,
          pch = 21, ret = FALSE, ...)
```

Arguments

obj	An MRexperiment object with count data.
otuIndex	A list of the otus with the same annotation.
classIndex	A list of the samples in their respective groups.
norm	Whether or not to normalize the counts.
no	Which of the otuIndex to plot.
factor	Factor value for jitter
pch	Standard pch value for the plot command.
jitter	Boolean to jitter the count data or not.
ret	Boolean to return the observed data that would have been plotted.
...	Additional plot arguments.

Value

NA

See Also

[cumNorm](#)

Examples

```
data(mouseData)
classIndex=list(controls=which(pData(mouseData)$diet=="BK"))
classIndex$cases=which(pData(mouseData)$diet=="Western")
otuIndex = grep("Strep", fData(mouseData)$fdata)
otuIndex=otuIndex[order(rowSums(MRcounts(mouseData)[otuIndex,]),decreasing=TRUE)]
plotGenus(mouseData,otuIndex,classIndex,xlab="OTU log-normalized counts",no=1:2,xaxt="n",norm=FALSE,ylab="Strep")
lablist<-rep(c("Controls","Cases"),times=2)
axis(1, at=seq(1,4,by=1), labels = lablist)
```

plotMRheatmap *Basic heatmap plot function for normalized counts.*

Description

This function plots a heatmap of the "n" features with greatest variance across rows.

Usage

```
plotMRheatmap(obj,n,trials,log=TRUE,norm=TRUE,...)
```

Arguments

obj	A MRexperiment object with count data.
n	The number of features to plot
trials	A vector of clinical information for.
log	Whether or not to log transform the counts.
norm	Whether or not to normalize the counts.
...	Additional plot arguments.

Value

NA

See Also

[cumNormMat](#)

Examples

```
data(mouseData)
trials = pData(mouseData)$diet
plotMRheatmap(obj=mouseData,n=200,trials=trials,cexRow = 0.4,cexCol = 0.4,trace="none")
```

plotOTU *Basic plot function of the raw or normalized data.*

Description

This function plots the abundance of a particular OTU by class. The function uses the estimated posterior probabilities to make technical zeros transparent.

Usage

```
plotOTU(obj, otu, classIndex, norm = TRUE,
        factor = 1, pch = 21, jitter = TRUE, ret = FALSE, ...)
```

Arguments

obj	A MRExperiment object with count data.
otu	The row number/OTU to plot.
classIndex	A list of the samples in their respective groups.
norm	Whether or not to normalize the counts.
factor	Factor value for jitter.
pch	Standard pch value for the plot command.
jitter	Boolean to jitter the count data or not.
ret	Boolean to return the observed data that would have been plotted.
...	Additional plot arguments.

Value

NA

See Also[cumNorm](#)**Examples**

```

data(mouseData)
classIndex=list(controls=which(pData(mouseData)$diet=="BK"))
classIndex$cases=which(pData(mouseData)$diet=="Western")
# you can specify whether or not to normalize, and to what level
plotOTU(mouseData,otu=9083,classIndex,xlab="OTU log-normalized counts",norm=FALSE,xaxt="n",main="9083 feature ab
lablist<- c("Controls","Cases")
axis(1, at=seq(1,2,by=1), labels = lablist)

```

posterior.probs *Access the posterior probabilities that results from analysis*

Description

Accessing the posterior probabilities following a run through [fitZig](#)

Usage

```

## S4 method for signature 'MRExperiment'
posterior.probs(obj)

```

Arguments

obj	a MRExperiment object.
-----	------------------------

Author(s)

Joseph N. Paulson, jpaulson@umiacs.umd.edu

Examples

```
#load(lungDataset)
#head(posterior.probs(lungData))
```

zigControl

Settings for the fitZig function

Description

Settings for the fitZig function

Usage

```
zigControl(tol = 1e-04, maxit = 10, verbose = TRUE)
```

Arguments

tol	The tolerance for the difference in negative log likelihood estimates for a feature to still be active.
maxit	The maximum number of iterations for the expectation-maximization algorithm.
verbose	Whether to display iterative step summary statistics or not.

Value

The value for the tolerance, maximum no. of iterations, and the verbose warning.

Note

[fitZig](#) makes use of zigControl.

See Also

[fitZig](#) [cumNorm](#) [plotOTU](#)

Examples

```
control = zigControl(tol=1e-10,maxit=10,verbose=FALSE)
```

Index

[,MRexperiment-method (MRexperiment), 20

aggregateM, 2

counts, 20

cumNorm, 3, 4, 8, 10, 20, 21, 27, 29, 30

cumNormMat, 3, 20, 26, 28

cumNormStat, 3, 4, 20

doCountMStep, 5

doEStep, 6

doZeroMStep, 6

exportMat, 7

exportMatrix (exportMat), 7

exportStats, 8

expSummary, 9

expSummary,MRexperiment-method
(expSummary), 9

fitZig, 3–7, 9, 11–14, 18, 21, 22, 24, 29, 30

genusPlot (plotGenus), 27

getCountDensity, 10

getEpsilon, 11

getNegativeLogLikelihoods, 12

getPi, 12

getZ, 13

isItStillActive, 14

libSize, 14, 20

libSize,MRexperiment-method (libSize),
14

load_meta, 15, 16, 17

load_metaQ, 16

load_phenoData, 15, 16, 16

lungData, 17

metagenomicLoader (load_meta), 15

mouseData, 17

MRcoefs, 18, 22, 24

MRcounts, 19

MRcounts,MRexperiment-method
(MRcounts), 19

MRexperiment, 20

MRexperiment-class (MRexperiment), 20

MRfisher, 21, 22

MRfulltable, 22

MRtable, 18, 22, 23

newMRexperiment, 20, 24

normFactors, 20, 25

normFactors,MRexperiment-method
(normFactors), 25

p.adjust, 18, 22, 23

phenoData (load_phenoData), 16

plotCorr, 26

plotGenus, 27

plotMRheatmap, 28

plotOTU, 28, 30

posterior.probs, 20, 29

posterior.probs,MRexperiment-method
(posterior.probs), 29

qiimeLoader (load_metaQ), 16

quantile, 8

settings (zigControl), 30

zigControl, 10, 30