Package 'metagenomeSeq'

April 5, 2014

Title Statistical analysis for sparse high-throughput sequencing

Version 1.4.2

Date 2013-10-06

Author Joseph Nathaniel Paulson, Mihai Pop, Hector Corrada Bravo

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

Description metagenomeSeq is designed to determine features (be it Operational Taxanomic Unit (OTU), species, etc.) that are differentially abundant between two or more groups of multiple samples. metagenomeSeq 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(>= 3.0), Biobase, limma, matrixStats, methods, RColorBrewer,gplots

Suggests annotate, knitr

biocViews Bioinformatics, DifferentialExpression, Metagenomics,Visualization

Collate 'aggregateM.R' 'allClasses.R' 'cumNorm.R' 'cumNormStat.R' 'cumNormMat.R' 'doCountMStep.R' 'doZeroMStep.R' 'doEStep.R' 'exportMat.R' 'exportStats.R' 'fitZig.R' 'getEpsilon.R' 'getCountDensity.R' 'getNegativeLogLikelihoods.R' 'getPi.R' 'getZ.R' 'isItStillActive.R' 'load_meta.R' 'load_metaQ.R' 'load_phenoData.R' 'MRtable.R' 'MRcoefs.R' 'MRfisher.R' 'MRfulltable.R' 'plotMRheatmap.R' 'plotCorr.R' 'plotOTU.R' 'plotOrd.R' 'plotRare.R' 'plotGenus.R' 'zigControl.R'

VignetteBuilder knitr

URL http://cbcb.umd.edu/software/metagenomeSeq

R topics documented:

metagenomeSeq-package *Statistical analysis for sparse high-throughput sequencing*

Description

metagenomeSeq is designed to determine features (be it Operational Taxanomic Unit (OTU), species, etc.) that are differentially abundant between two or more groups of multiple samples. metagenomeSeq 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.

A user's guide is available, and can be opened by typing `vignette("metagenomeSeq")`

The metagenomeSeq package implements novel normalization and statistical methodology in the following papers.

Author(s)

Paulson, JN <jpaulson@umiacs.umd.edu>; Pop, M; Corrada Bravo, H

References

Paulson, Joseph N., O. Colin Stine, Hector Corrada Bravo, and Mihai Pop. "Differential abundance analysis for microbial marker-gene surveys." Nature methods (2013).

aggregateM *Aggregates counts by a particular classification.*

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 taxanomic levels.

cumNorm *Cumulative sum scaling factors.*

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 *Cumulative sum scaling factors.*

Description

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

Usage

```
cumNormMat(obj, p = cumNormStat(obj), sl = 1000)
```

Arguments

obj	A MRexperiment object.
p	The pth quantile.
sl	The value to scale by (default=1000).

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 *Cumulative sum scaling percentile selection*

Description

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

Usage

```
cumNormStat(obj, qFlag = TRUE, pFlag = FALSE, rel = 0.1,
  ...)
```

Arguments

obj	A list with count data
qFlag	Flag to either calculate the proper percentile using a step-wise or triangular approximation of the sample count distribution (default step-wise).
pFlag	Plot the median difference quantiles
rel	Cutoff for the relative difference from one median difference from the reference to the next
...	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)
```

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

z	Matrix (m x n) of estimate responsibilities (probabilities that a count comes from a spike distribution at 0).
y	Matrix (m x n) of count observations.
mmCount	Model matrix for the count distribution.
stillActive	Boolean vector of size M, indicating whether a feature converged or not.
fit2	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 $f_zig(y_ij = pi_j(S_j)*f_0(y_ij) +(1-pi_j (S_j)) * f_count(y_ij;mu_i,sigma_i^2)$. The log-likelihood in this extended model is $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (s_j))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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 = fracpi_j cdot I_0(y_ijpi_j cdot I_0(y_ij + (1-pi_j) cdot f_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 $f_zig(y_ij = pi_j(S_j) cdot f_0(y_ij) +(1-pi_j (S_j))cdot f_count(y_ij;mu_i,sigma_i^2)$. The log-likelihood in this extended model is $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (sj))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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. $$ pi_j = sum_i^M frac1Mz_ij $$ 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 $f_zig(y_ij = pi_j(S_j) cdot f_0(y_ij) +(1-pi_j (S_j))cdot f_count(y_ij;mu_i,sigma_i^2)$. The log-likelihood in this extended model is $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (sj))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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 sum(zeroIndices).

See Also

fitZig

---

exportMat *export the normalized eSet dataset as a matrix.*

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(obj, log = TRUE, norm = TRUE,
  output = "~/Desktop/matrix.tsv")
```

Arguments

obj	A MRexperiment object with count data or matrix.
log	Whether or not to log transform the counts - if MRexperiment object.
norm	Whether or not to normalize the counts - if MRexperiment object.
output	Output file name

Value

NA

See Also

cumNorm

Examples

```
# see vignette
```

---

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, the number identified features, and library size (depth of coverage).

Usage

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

Arguments

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

Value

None.

See Also

cumNorm quantile

Examples

```
# see vignette
```

`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

```
expSummary(obj)
```

Arguments

`obj`	a `MRexperiment` object.

Author(s)

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

Examples

```
data(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 $f_zig(y_ij = pi_j(S_j)*f_0(y_ij) +(1-pi_j (S_j)) * f_count(y_ij; mu_i, sigma_i^2)$. The log-likelihood in this extended model is: $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij) log (1-pi_j (s_j))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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)$SampleType)
lungTrim = lungData[,-k]
k = which(rowSums(MRcounts(lungTrim)>0)<30)
cumNorm(lungTrim)
lungTrim = lungTrim[-k,]
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
settings = zigControl(maxit=1,verbose=FALSE)
fit = fitZig(obj = lungTrim,mod=mod,control=settings)
```

---

getCountDensity *Compute the value of the count density function from the count model residuals.*

Description

Calculate density values from a normal: $f(x) = 1/(sqrt (2 pi) sigma) e^-((x - mu)^2/(2 sigma^2))$. Maximum-likelihood estimates are approximated using the EM algorithm where we treat mixture membership $deta_ij$ = 1 if y_ij is generated from the zero point mass as latent indicator variables. The density is defined as $f_zig(y_ij = pi_j(S_j) cdot f_0(y_ij) +(1-pi_j (S_j))cdot f_count(y_ij;mu_i,sigma_i^2)$. The log-likelihood in this extended model is $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (sj))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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 *Calculate the relative difference between iterations of the negative log-likelihoods.*

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 $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (sj))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data)$.

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 $(1-delta_ij) log f_count(y;mu_i,sigma_i^2)+delta_ij log pi_j(s_j)+(1-delta_ij)log (1-pi_j (sj))$. The responsibilities are defined as $z_ij = pr(delta_ij=1 | data and current values)$.

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

---

getZ *Calculate the current Z estimate responsibilities (posterior probabilities)*

Description

Calculate the current Z estimate responsibilities (posterior probabilities)

Usage

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

Arguments

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

eps	Vector of size M (features) representing the relative difference between the new nll and old nll.
tol	The threshold tolerance for the difference
stillActive	A vector of size M booleans saying if a feature is still active or not.
stillActiveNLL	A vector of size M recording the negative log-likelihoods of the various features, updated for those still active.
nll	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 or depth of coverage. It is used by fitZig.

Usage

```
libSize(obj)
```

Arguments

obj	a MRexperiment object.

Author(s)

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

Examples

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

---

load_meta *Load a count dataset associated with a study.*

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

```
dataDirectory <- system.file("extdata", package="metagenomeSeq")
lung = load_meta(file.path(dataDirectory,"CHK_NAME.otus.count.csv"))
```

load_metaQ *Load a count dataset associated with a study set up in a Qiime format.*

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

```
# see vignette
```

load_phenoData *Load a clinical/phenotypic dataset associated with a study.*

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

```
# see vignette
```

---

lungData *OTU abundance matrix of samples from a smoker/non-smoker study*

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 *OTU abundance matrix of mice samples from a diet longitudinal study*

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, uniqueNames = FALSE,
  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.
`uniqueNames`	Number the various taxa.
`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, 0,1,2,3. 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. 3: the sort is ordered by the p-value of the 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

```
MRcounts(obj, norm = FALSE, log = FALSE, sl = 1000)
```

Arguments

obj	a MRexperiment object.
norm	logical indicating whether or not to return normalized counts.
log	TRUE/FALSE whether or not to log_2 transform scale.
sl	The value to scale by (default=1000).

Author(s)

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

Examples

```
data(lungData)
head(MRcounts(lungData))
```

MRexperiment *Class "MRexperiment" – a modified eSet object for the data from high-throughput sequencing experiments*

Description

This is the main class for metagenomeSeq.

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 metagenomeSeq. The class extends eSet and provides additional slots which are populated during the analysis pipeline.

MRexperiment dataset are created with calls to newMRexperiment. MRexperiment datasets contain raw count matrices (integers) accessible through MRcounts. Similarly, normalized count matrices can be accessed (following normalization) through MRcounts 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, thres = 0)
```

Arguments

obj	A MRexperiment object with a count matrix, or a simple count matrix.
cl	Group comparison
thres	Threshold for defining presence/absence.

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 *Table of top microbial marker gene from linear model fit including sequence information*

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, uniqueNames = FALSE,
  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.
uniqueNames	Number the various taxa.
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. 3: the sort is ordered by the p-value of the 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 *Table of top microbial marker gene from linear model fit including sequence information*

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, uniqueNames = FALSE,
  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.
`uniqueNames`	Number the various taxa.
`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. 3: the sort is ordered by the p-value of the 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

```
normFactors(obj)
```

Arguments

obj	a MRexperiment object.

Author(s)

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

Examples

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

plotCorr *Basic correlation plot function for normalized or unnormalized counts.*

Description

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

Usage

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

Arguments

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

Value

NA

See Also

cumNormMat

Examples

```
data(mouseData)
plotCorr(obj=mouseData,n=200,cexRow = 0.4,cexCol = 0.4,trace="none",dendrogram="none",
         col = colorRampPalette(brewer.pal(9, "RdBu"))(50))
```

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, log = TRUE,
  norm = TRUE, no = 1:length(otuIndex), labs = TRUE,
  xlab = NULL, ylab = NULL, jitter = TRUE,
  jitter.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.
log	Whether or not to log transform the counts - if MRexperiment object.
norm	Whether or not to normalize the counts - if MRexperiment object.
no	Which of the otuIndex to plot.
jitter.factor	Factor value for jitter
pch	Standard pch value for the plot command.
labs	Whether to include group labels or not. (TRUE/FALSE)
xlab	xlabel for the plot.
ylab	ylabel for the plot.
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,no=1:2,xaxt="n",norm=FALSE,ylab="Strep normalized log(cpt)")
```

`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, log = TRUE, norm = TRUE, ...)
```

Arguments

`obj`	A MRexperiment object with count data.
`n`	The number of features to plot
`log`	Whether or not to log transform the counts - if MRexperiment object.
`norm`	Whether or not to normalize the counts - if MRexperiment object.
`...`	Additional plot arguments.

Value

NA

See Also

`cumNormMat`

Examples

```
data(mouseData)
trials = pData(mouseData)$diet
heatmapColColors=brewer.pal(12,"Set3")[as.integer(factor(trials))];
heatmapCols = colorRampPalette(brewer.pal(9, "RdBu"))(50)
plotMRheatmap(obj=mouseData,n=200,cexRow = 0.4,cexCol = 0.4,trace="none",
              col = heatmapCols,ColSideColors = heatmapColColors)
```

plotOrd *Plot of either PCA or MDS coordinates for the distances of normalized or unnormalized counts.*

Description

This function plots the PCA / MDS coordinates for the "n" features of interest. Potentially uncovering batch effects or feature relationships.

Usage

```
plotOrd(obj, tran = FALSE, comp = 1:2, log = TRUE,
  norm = TRUE, usePCA = TRUE, useDist = FALSE,
  dist.method = "euclidian", ret = FALSE, ntop = NULL,
  ...)
```

Arguments

obj	A MRexperiment object or count matrix.
tran	Transpose the matrix.
comp	Which components to display
usePCA	TRUE/FALSE whether to use PCA or MDS coordinates (TRUE is PCA).
useDist	TRUE/FALSE whether to calculate distances.
dist.method	If useDist==TRUE, what method to calculate distances.
log	Whether or not to log the counts - if MRexperiment object.
norm	Whether or not to normalize the counts - if MRexperiment object.
ret	Whether or not to output the coordinates.
ntop	Number of features to make use of in calculating your distances.
...	Additional plot arguments.

Value

NA

See Also

cumNormMat

Examples

```
data(mouseData)
cl = pData(mouseData)[,3]
plotOrd(mouseData,tran=TRUE,useDist=TRUE,pch=21,bg=factor(cl),usePCA=FALSE)
```

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, log = TRUE, norm = TRUE,
  jitter.factor = 1, pch = 21, labs = TRUE, xlab = NULL,
  ylab = NULL, 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.
log	Whether or not to log transform the counts - if MRexperiment object.
norm	Whether or not to normalize the counts - if MRexperiment object.
jitter.factor	Factor value for jitter.
pch	Standard pch value for the plot command.
labs	Whether to include group labels or not. (TRUE/FALSE)
xlab	xlabel for the plot.
ylab	ylabel for the plot.
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,norm=FALSE,main="9083 feature abundances")
```

plotRare *Plot of rarefaction effect*

Description

This function plots the number of observed features vs. the depth of coverage.

Usage

```
plotRare(obj, cl = NULL, ret = FALSE, ...)
```

Arguments

obj	A MRexperiment object with count data or matrix.
cl	Vector of classes for various samples.
ret	True/False, return the number of features and the depth of coverage as a vector.
...	Additional plot arguments.

Value

NA

See Also

plotOrd, plotMRheatmap, plotCorr, plotOTU, plotGenus

Examples

```
data(mouseData)
cl = factor(pData(mouseData)[,3])
res = plotRare(mouseData,cl=cl,ret=TRUE,pch=21,bg=cl)
tmp=lapply(levels(cl), function(lv) lm(res[,"ident"]~res[,"libSize"]-1, subset=cl==lv))
for(i in 1:length(levels(cl))){
   abline(tmp[[i]], col=i)
}
legend("topleft", c("Diet 1","Diet 2"), text.col=c(1,2),box.col=NA)
```

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

Description

Accessing the posterior probabilities following a run through `fitZig`

Usage

```
posterior.probs(obj)
```

Arguments

`obj`	a `MRexperiment` object.

Author(s)

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

Examples

```
# see vignette
```

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 remain 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