Package ‘metagenomeSeq’

June 28, 2022

Title Statistical analysis for sparse high-throughput sequencing
Version 1.38.0
Date 2019-07-12
Author Joseph Nathaniel Paulson, Nathan D. Olson, Domenick J. Braccia,
       Justin Wagner, Hisham Talukder, Mihai Pop, Hector Corrada Bravo
Maintainer Joseph N. Paulson <jpaulson@jimmy.harvard.edu>
Description metagenomeSeq 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. 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, glmnet, methods, RColorBrewer
Suggests annotate, BiocGenerics, biomformat, knitr, gss, testthat (>=
       0.8), vegan, interactiveDisplay, IHW
Imports parallel, matrixStats, foreach, Matrix, gplots, graphics,
grDevices, stats, utils, Wrench
VignetteBuilder knitr
URL https://github.com/nosson/metagenomeSeq/
BugReports https://github.com/nosson/metagenomeSeq/issues
biocViews ImmunoOncology, Classification, Clustering,
       GeneticVariability, DifferentialExpression, Microbiome,
       Metagenomics, Normalization, Visualization, MultipleComparison,
       Sequencing, Software
RoxygenNote 7.1.0
git_url https://git.bioconductor.org/packages/metagenomeSeq
git_branch RELEASE_3_15
git_last_commit 50be9db
R topics documented:

- metagenomeSeq-package
- aggregateBySample
- aggregateByTaxonomy
- biom2MRexperiment
- calcNormFactors
- calcPosComponent
- calcShrinkParameters
- calcStandardError
- calculateEffectiveSamples
- calcZeroAdjustment
- calcZeroComponent
- correctIndices
- correlationTest
- cumNorm
- cumNormMat
- cumNormStat
- cumNormStatFast
- doCountMStep
- doEStep
- doZeroMStep
- exportMat
- exportStats
- expSummary
- extractMR
- filterData
- fitDO
- fitFeatureModel
- fitFeatureModelResults-class
- fitLogNormal
- fitMultipleTimeSeries
- fitPA
- fitSSTimeSeries
- fitTimeSeries
- fitZeroLogNormal
- fitZig
- fitZigResults-class
- getCountDensity
- getEpsilon
- getNegativeLogLikelihoods
- getPi
- getZ
- isItStillActive
R topics documented:

libSize ................................. 38
libSize<- .............................. 39
loadBiom .............................. 40
loadMeta .............................. 40
loadMetaQ ............................. 41
loadPhenoData ........................ 42
lungData .............................. 42
makeLabels ............................ 43
mergeMRexperiments .................. 44
mergeTable ............................ 44
metagenomeSeq-deprecated .......... 45
mouseData ............................ 45
MRecoins .............................. 46
MRcounts .............................. 47
MRexperiment ......................... 48
MRexperiment2biom ................... 49
MRfulltable ........................... 50
MRihw ................................. 52
MRihw,fitFeatureModelResults-method 52
MRihw,fitZigResults-method ........ 53
MRtable ............................... 53
newMRexperiment ..................... 55
normFactors ......................... 56
normFactors<- ....................... 57
plotBubble ........................... 57
plotClassTimeSeries ................. 59
plotCorr ............................. 60
plotFeature .......................... 61
plotGenus ........................... 62
plotMRheatmap ....................... 63
plotOrd .............................. 64
plotOTU .............................. 65
plotRare ............................. 67
plotTimeSeries ....................... 68
posteriorProbs ....................... 69
returnAppropriateObj ............... 70
ssFit ................................. 70
ssIntervalCandidate ................. 71
ssPerm ............................... 72
ssPermAnalysis ...................... 73
trapez ............................... 74
ts2MRexperiment ..................... 74
uniqueFeatures ...................... 76
wrenchNorm ........................... 76
zigControl ........................... 77

Index 79
**metagenomeSeq-package**  
*Statistical analysis for sparse high-throughput sequencing*

**Description**

metagenomeSeq 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. 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 <j paulson@umiacs.umd.edu>; Pop, M; Corrada Bravo, H

**References**


---

**aggregateBySample**  
*Aggregates a MRexperiment object or counts matrix to by a factor.*

**Description**

Using the phenoData information in the MRexperiment, calling aggregateBySample on a MRexperiment and a particular phenoData column (i.e. 'diet') will aggregate counts using the aggfun function (default rowMeans). Possible aggfun alternatives include rowMeans and rowMedians.

**Usage**

```r
aggregateBySample(obj, fct, aggfun = rowMeans, out = "MRexperiment")
	aggSamp(obj, fct, aggfun = rowMeans, out = "MRexperiment")
```

**Arguments**

- `obj`: A MRexperiment object or count matrix.
- `fct`: phenoData column name from the MRexperiment object or if count matrix object a vector of labels.
- `aggfun`: Aggregation function.
- `out`: Either 'MRexperiment' or 'matrix'
aggregateByTaxonomy

Value

An aggregated count matrix or MRexperiment object where the new pData is a vector of 'fct' levels.

Examples

data(mouseData)
aggregateBySample(mouseData[1:100,],fct="diet",aggfun=rowSums)
# not run
# aggregateBySample(mouseData,fct="diet",aggfun=matrixStats::rowMedians)
# aggSamp(mouseData,fct='diet',aggfun=rowMaxs)

aggregateByTaxonomy

Aggregates a MRexperiment object or counts matrix to a particular level.

Description

Using the featureData information in the MRexperiment, calling aggregateByTaxonomy on a MRexperiment and a particular featureData column (i.e. 'genus') will aggregate counts to the desired level using the aggfun function (default colSums). Possible aggfun alternatives include colMeans and colMedians.

Usage

aggregateByTaxonomy(
  obj,
  lvl,
  alternate = FALSE,
  norm = FALSE,
  log = FALSE,
  aggfun = colSums,
  sl = 1000,
  featureOrder = NULL,
  returnFullHierarchy = TRUE,
  out = "MRexperiment"
)

aggTax(
  obj,
  lvl,
  alternate = FALSE,
  norm = FALSE,
  log = FALSE,
  aggfun = colSums,
  sl = 1000,
  featureOrder = NULL,
Arguments

obj A MRexperiment object or count matrix.
lvl featureData column name from the MRexperiment object or if count matrix object a vector of labels.
alternate Use the rowname for undefined OTUs instead of aggregating to "no_match".
norm Whether to aggregate normalized counts or not.
log Whether or not to log2 transform the counts - if MRexperiment object.
aggfun Aggregation function.
sl scaling value, default is 1000.
featureOrder Hierarchy of levels in taxonomy as fData colnames
returnFullHierarchy Boolean value to indicate return single column of fData or all columns of hierarchy
out Either 'MRexperiment' or 'matrix'

Value

An aggregated count matrix.

Examples

data(mouseData)
aggregateByTaxonomy(mouseData[1:100,],lvl="class",norm=TRUE,aggfun=colSums)
# not run
# aggregateByTaxonomy(mouseData,lvl="class",norm=TRUE,aggfun=colMedians)
# aggTax(mouseData,lvl='phylum',norm=FALSE,aggfun=colSums)

biom2MRexperiment Biom to MRexperiment objects

Description

Wrapper to convert biom files to MRexperiment objects.

Usage

biom2MRexperiment(obj)
calcNormFactors

Arguments

obj The biom object file.

Value

A MRexperiment object.

See Also

loadMeta loadPhenoData newMRexperiment loadBiom

Examples

library(biomformat)
rich_dense_file = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
x = biomformat::read_biom(rich_dense_file)
biom2MRexperiment(x)

calcNormFactors Cumulative sum scaling (css) normalization factors

Description

Return a vector of the sum up to and including a quantile.

Usage

calcNormFactors(obj, p = cumNormStatFast(obj))

Arguments

obj An MRexperiment object or matrix.

p The pth quantile.

Value

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

See Also

fitZig cumNormStatFast cumNorm

Examples

data(mouseData)
head(calcNormFactors(mouseData))
calcPosComponent  Positive component

Description
Fit the positive (log-normal) component

Usage
calcPosComponent(mat, mod, weights)

Arguments
mat       A matrix of normalized counts
mod       A model matrix
weights   Weight matrix for samples and counts

See Also
fitZeroLogNormal fitFeatureModel

calcShrinkParameters  Calculate shrinkage parameters

Description
Calculate the shrunken variances and variance of parameters of interest across features.

Usage
calcShrinkParameters(fit, coef, mins2, exclude = NULL)

Arguments
fit         A matrix of fits as outputted by calcZeroComponent or calcPosComponent
coef        Coefficient of interest
mins2       minimum variance estimate
exclude     Vector of features to exclude when shrinking

See Also
fitZeroLogNormal fitFeatureModel
calcStandardError

Description

Calculate the zero-inflated log-normal statistic’s standard error

Usage

calcStandardError(mod, fitln, fitzero, coef = 2, exclude = NULL)

Arguments

mod          The zero component model matrix
fitln        A matrix with parameters from the log-normal fit
fitzero      A matrix with parameters from the logistic fit
coef         Coefficient of interest
exclude      List of features to exclude

See Also

fitZeroLogNormal fitFeatureModel

calculateEffectiveSamples

Estimated effective samples per feature

Description

Calculates the number of estimated effective samples per feature from the output of a fitZig run. The estimated effective samples per feature is calculated as the sum_{1}^{n} (n = number of samples) 1-z_i where z_i is the posterior probability a feature belongs to the technical distribution.

Usage

calculateEffectiveSamples(obj)

Arguments

obj          The output of fitZig run on a MRexperiment object.

Value

A list of the estimated effective samples per feature.
See Also

`fitZig MRcoefs MRfulltable`

calcZeroAdjustment Calculate the zero-inflated component's adjustment factor

Description

Calculate the log ratio of average marginal probabilities for each sample having a positive count. This becomes the adjustment factor for the log fold change.

Usage

calcZeroAdjustment(fitln, fitzero, mod, coef, exclude = NULL)

Arguments

- `fitln`: A matrix with parameters from the log-normal fit
- `fitzero`: A matrix with parameters from the logistic fit
- `mod`: The zero component model matrix
- `coef`: Coefficient of interest
- `exclude`: List of features to exclude

See Also

`fitZeroLogNormal fitFeatureModel`

calcZeroComponent Zero component

Description

Fit the zero (logisitic) component

Usage

calcZeroComponent(mat, mod, weights)

Arguments

- `mat`: A matrix of normalized counts
- `mod`: A model matrix
- `weights`: Weight matrix for samples and counts

See Also

`fitZeroLogNormal fitFeatureModel`
correctIndices

Calculate the correct indices for the output of correlationTest

Description

Consider the upper triangular portion of a matrix of size nxn. Results from the correlationTest are output as the combination of two vectors, correlation statistic and p-values. The order of the output is 1vs2, 1vs3, 1vs4, etc. The correctIndices returns the correct indices to fill a correlation matrix or correlation-pvalue matrix.

Usage

correctIndices(n)

Arguments

n The number of features compared by correlationTest (nrow(mat)).

Value

A vector of the indices for an upper triangular matrix.

See Also

correlationTest

Examples

data(mouseData)
mat = MRcounts(mouseData)[55:60,]
cors = correlationTest(mat)
ind = correctIndices(nrow(mat))

cormat = as.matrix(dist(mat))
cormat[cormat>0] = 0
cormat[upper.tri(cormat)][ind] = cors[,1]
table(cormat[1,-1] - cors[1:5,1])
correlationTest  

**Correlation of each row of a matrix or MReperiment object**

**Description**

Calculates the (pairwise) correlation statistics and associated p-values of a matrix or the correlation of each row with a vector.

**Usage**

```r
correlationTest(
  obj,
  y = NULL,
  method = "pearson",
  alternative = "two.sided",
  norm = TRUE,
  log = TRUE,
  cores = 1,
  override = FALSE,
  ...
)
```

**Arguments**

- `obj` A MReperiment object or count matrix.
- `y` Vector of length ncol(obj) to compare to.
- `method` One of 'pearson', 'spearman', or 'kendall'.
- `alternative` Indicates the alternative hypothesis and must be one of 'two.sided', 'greater' (positive) or 'less'(negative). You can specify just the initial letter.
- `norm` Whether to aggregate normalized counts or not - if MReperiment object.
- `log` Whether or not to log2 transform the counts - if MReperiment object.
- `cores` Number of cores to use.
- `override` If the number of rows to test is over a thousand the test will not commence (unless override==TRUE).
- `...` Extra parameters for mclapply.

**Value**

A matrix of size choose(number of rows, 2) by 2. The first column corresponds to the correlation value. The second column the p-value.

**See Also**

`correctIndices`
**Examples**

```r
data(mouseData)
cors = correlationTest(mouseData[1:10,], norm=FALSE, log=FALSE)
head(cors)

mat = MRcounts(mouseData)[1:10,]
cormat = as.matrix(dist(mat)) # Creating a matrix
cormat[cormat>0] = 0 # Creating an empty matrix
ind = correctIndices(nrow(mat))
cormat[upper.tri(cormat)][ind] = cors[,1]
table(cormat[1,-1] - cors[1:9,1])

# Correlation of raw counts with a vector (library size in this case)
data(mouseData)
cors = correlationTest(mouseData[1:10,], libSize(mouseData), norm=FALSE, log=FALSE)
head(cors)
```

---

**cumNorm**

*Cumulative sum scaling normalization*

**Description**

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

**Usage**

```r
cumNorm(obj, p = cumNormStatFast(obj))
```

**Arguments**

- `obj`: An MRexperiment object.
- `p`: The pth quantile.

**Value**

Object with the normalization factors stored as a vector of the sum up to and including a sample’s pth quantile.

**See Also**

`fitZig`, `cumNormStat`

**Examples**

```r
data(mouseData)
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
\[
cumNormMat(obj, p = \text{cumNormStatFast}(obj), sl = 1000)
\]

Arguments
- **obj**: A matrix or 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. cumNormStat might be deprecated one day. Deviates from methods in Nature Methods paper by making use row means for generating reference.

Usage
\[
cumNormStat(obj, qFlag = \text{TRUE}, pFlag = \text{FALSE}, rel = 0.1, \ldots)
\]
cumNormStatFast

Arguments

- **obj**: A matrix or MRexperiment object.
- **qFlag**: Flag to either calculate the proper percentile using R’s step-wise quantile function or approximate function.
- **pFlag**: Plot the relative difference of the median deviance from the reference.
- **rel**: Cutoff for the relative difference from one median difference from the reference to the next.
- **...**: Applicable if pFlag == TRUE. Additional plotting parameters.

Value

Percentile for which to scale data

See Also

fitZig, cumNorm, cumNormStatFast

Examples

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

---

cumNormStatFast  Cumulative sum scaling percentile selection

Description

Calculates the percentile for which to sum counts up to and scale by. Faster version than available in cumNormStat. Deviates from methods described in Nature Methods by making use of ro means for reference.

Usage

cumNormStatFast(obj, pFlag = FALSE, rel = 0.1, ...)

Arguments

- **obj**: A matrix or MRexperiment object.
- **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. Additional plotting parameters.
doCountMStep

Value
Percentile for which to scale data

See Also
fitZig, cumNorm, cumNormStat

Examples
```
data(mouseData)
p = round(cumNormStatFast(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, dfMethod = "modified")
```

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.
- **dfMethod**: Either 'default' or 'modified' (by responsibilities)

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 $f_{\text{zig}}(y_{ij} = \pi_j(S_j)\phi_0(y_{ij}) +(1-\pi_j (S_j)) \times f_{\text{count}}(y_{ij};\mu_i,\sigma_i^2)$.

The log-likelihood in this extended model is $(1-\delta_{ij}) \log f_{\text{count}}(y_{ij};\mu_i,\sigma_i^2) + \delta_{ij} \log \pi_j(s_j) + (1-\delta_{ij}) \log (1-\pi_j (s_j))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij}=1 | \text{data})$. 
**doEStep**

**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 $z_{ij} = \frac{\pi_j}{\pi_j I_0(y_{ij}) + (1-\pi_j)I_0(y_{ij} + (1-\pi_j)f_{count}(y_{ij})}$.

**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 $f_{zrig}(y_{ij} = \pi_j(S_j)I_0(y_{ij}) + (1-\pi_j(S_j))I_0(y_{ij} + (1-\pi_j)f_{count}(y_{ij};mu_i,\sigma_i^2)$. The log-likelihood in this extended model is $f_{1-\delta_{ij}} log f_{count}(y_{ij};mu_i,\sigma_i^2)$ + $\delta_{ij} log \pi_j(s_j)$ + $(1-\delta_{ij}) log (1-\pi_j(s_j))$. The responsibilities are defined as $z_{ij} = pr(\delta_{ij}=1 | data)$.

**Value**

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. $\pi_j = \sum_{i=1}^{M} \frac{1}{M} z_{ij}$ 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 $z_{ij} = \Pr(\delta_{ij} = 1 | \text{data})$.

**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 sum(zeroIndices).

**See Also**

`fitZig`
Usage

```r
exportMat(
  obj,
  log = TRUE,
  norm = TRUE,
  sep = "\t",
  file = "~/Desktop/matrix.tsv"
)
```

Arguments

- `obj`: A MRexperiment object or count matrix.
- `log`: Whether or not to log transform the counts - if MRexperiment object.
- `norm`: Whether or not to normalize the counts - if MRexperiment object.
- `sep`: Separator for writing out the count matrix.
- `file`: Output file name.

Value

NA

See Also

cumNorm

Examples

```r
data(lungData)
dataDirectory <- system.file("extdata", package="metagenomeSeq")
exprotMat(lungData[,1:5], file=file.path(dataDirectory,"tmp.tsv"))
head(read.csv(file=file.path(dataDirectory,"tmp.tsv"), sep="\t"))
```

---

**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

```r
exportStats(obj, p = cumNormStat(obj), file = "~/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.
file Output file name.

Value

None.

See Also
cumNorm quantile

Examples

data(lungData)
dataDirectory <- system.file("extdata", package="metagenomeSeq")
exportStats(lungData[,1:5],file=file.path(dataDirectory,"tmp.tsv"))
head(read.csv(file=file.path(dataDirectory,"tmp.tsv"),sep="\t"))

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.

Value

Experiment summary table

Author(s)

Joseph N. Paulson, j paulson@umiacs.umd.edu
extractMR

Examples

data(mouseData)
expSummary(mouseData)

extractMR

Description

Extract the essentials of an MRexperiment.

Usage

extractMR(obj)

Arguments

obj MRexperiment-class object.

Value

A list containing:

- counts : Count data
- librarySize : The column sums / library size / sequencing depth
- normFactors : The normalization scaling factors
- pheno : phenotype table
- feat : feature table

Examples

data(mouseData)
head(metagenomeSeq:::extractMR(mouseData))
filterData

Filter datasets according to no. features present in features with at least a certain depth.

Description

Filter the data based on the number of present features after filtering samples by depth of coverage. There are many ways to filter the object, this is just one way.

Usage

filterData(obj, present = 1, depth = 1000)

Arguments

- **obj**: A MRexperiment object or count matrix.
- **present**: Features with at least 'present' positive samples.
- **depth**: Samples with at least this much depth of coverage

Value

A MRexperiment object.

Examples

data(mouseData)
filterData(mouseData)

fitDO

Wrapper to calculate Discovery Odds Ratios on feature values.

Description

This function returns a data frame of p-values, odds ratios, lower and upper confidence limits for every row of a matrix. The discovery odds ratio is calculated as using Fisher’s exact test on actual counts. The test’s hypothesis is whether or not the discovery of counts for a feature (of all counts) is found in greater proportion in a particular group.

Usage

fitDO(obj, cl, norm = TRUE, log = TRUE, adjust.method = "fdr", cores = 1, ...)
fitFeatureModel

Computes differential abundance analysis using a zero-inflated log-normal model

Description

Wrapper to actually run zero-inflated log-normal model given a MRexperiment object and model matrix. User can decide to shrink parameter estimates.

Usage

fitFeatureModel(obj, mod, coef = 2, B = 1, szero = FALSE, spos = TRUE)
fitFeatureModelResults-class

Class "fitFeatureModelResults" – a formal class for storing results from a fitFeatureModel call

Description
This class contains all of the same information expected from a fitFeatureModel call, but it is defined in the S4 style as opposed to being stored as a list.
**fitLogNormal**

**Description**

Wrapper to perform the permutation test on the t-statistic. This is the original method employed by metastats (for non-sparse large samples). We include CSS normalization though (optional) and log2 transform the data. In this method the null distribution is not assumed to be a t-dist.

**Usage**

```r
fitLogNormal(obj, mod, useCSSoffset = TRUE, B = 1000, coef = 2, sl = 1000)
```

**Arguments**

- `obj` A MRexperiment object with count data.
- `mod` The model for the count distribution.
- `useCSSoffset` Boolean, whether to include the default scaling parameters in the model or not.
- `B` Number of permutations.
- `coef` The coefficient of interest.
- `sl` The value to scale by (default=1000).

**Value**

Call made, fit object from lmFit, t-statistics and p-values for each feature.

**Examples**

```r
# This is a simple demonstration
data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[-k]
k = which(rowSums(MRcounts(lungTrim)>0)<30)
lungTrim = cumNorm(lungTrim)
lungTrim = lungTrim[-k]
```
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
fit = fitLogNormal(obj = lungTrim, mod=mod, B=1)

fitMultipleTimeSeries (obj, lvl = NULL, B = 1, featureOrder = NULL, ...)

Arguments

obj metagenomeSeq MRexperiment-class object.
lvl Vector or name of column in featureData of MRexperiment-class object for aggregating counts (if not OTU level).
B Number of permutations to perform.
featureOrder Hierarchy of levels in taxonomy as fData colnames
... Options for fitTimeSeries, except feature.

Value

List of lists of matrices of time point intervals of interest, Difference in abundance area and p-value, fit, area permutations.

A list of lists for which each includes:

- timeIntervals - Matrix of time point intervals of interest, area of differential abundance, and p-value.
- data - Data frame of abundance, class indicator, time, and id input.
- fit - Data frame of fitted values of the difference in abundance, standard error estimates and timepoints interpolated over.
- perm - Differential abundance area estimates for each permutation.
- call - Function call.

See Also

cumNorm fitSSTimeSeries fitTimeSeries
Examples

```r
data(mouseData)
res = fitMultipleTimeSeries(obj=mouseData,lvl='phylum',class="status",
   id="mouseID",time="relativeTime",B=1)
```

**fitPA**

*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**

```r
fitPA(obj, cl, thres = 0, adjust.method = "fdr", cores = 1, ...)
```

**Arguments**

- `obj`: A MRexperiment object with a count matrix, or a simple count matrix.
- `cl`: Group comparison
- `thres`: Threshold for defining presence/absence.
- `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.
- `cores`: Number of cores to use.
- `...`: Extra parameters for makeCluster

**Value**

Matrix of odds ratios, p-values, lower and upper confidence intervals

**See Also**

`cumNorm` `fitZig` `fitDO` `fitMeta`

**Examples**

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

*Discover differentially abundant time intervals using SS-Anova*

**Description**

Calculate time intervals of interest using SS-Anova fitted models. Fitting is performed uses Smoothing Spline ANOVA (SS-Anova) to find interesting intervals of time. Given observations at different time points for two groups, fitSSTimeSeries calculates a function that models the difference in abundance between two groups across all time. Using permutations we estimate a null distribution of areas for the time intervals of interest and report significant intervals of time. Use of the function for analyses should cite: "Finding regions of interest in high throughput genomics data using smoothing splines" Talukder H, Paulson JN, Bravo HC. (In preparation)

**Usage**

```r
fitSSTimeSeries(
  obj, 
  formula, 
  feature, 
  class, 
  time, 
  id, 
  lvl = NULL, 
  include = c("class", "time:class"), 
  C = 0, 
  B = 1000, 
  norm = TRUE, 
  log = TRUE, 
  sl = 1000, 
  featureOrder = NULL, 
  ...
)
```

**Arguments**

- **obj**: metagenomeSeq MRexperiment-class object.
- **formula**: Formula for ssanova. Of the form: abundance ~ ... where ... includes any pData slot value.
- **feature**: Name or row of feature of interest.
- **class**: Name of column in phenoData of MRexperiment-class object for class membership.
- **time**: Name of column in phenoData of MRexperiment-class object for relative time.
- **id**: Name of column in phenoData of MRexperiment-class object for sample id.
- **lvl**: Vector or name of column in featureData of MRexperiment-class object for aggregating counts (if not OTU level).
include  Parameters to include in prediction.
C       Value for which difference function has to be larger or smaller than (default 0).
B       Number of permutations to perform
norm    When aggregating counts to normalize or not.
log     Log2 transform.
sl      Scaling value.
featureOrder  Hierarchy of levels in taxonomy as fData colnames
...     Options for ssanova

Value
List of matrix of time point intervals of interest, Difference in abundance area and p-value, fit, area
permutations, and call.
A list of objects including:
  • timeIntervals - Matrix of time point intervals of interest, area of differential abundance, and
p-value.
  • data - Data frame of abundance, class indicator, time, and id input.
  • fit - Data frame of fitted values of the difference in abundance, standard error estimates and
timepoints interpolated over.
  • perm - Differential abundance area estimates for each permutation.
  • call - Function call.

See Also
cumNorm ssFit ssIntervalCandidate ssPerm ssPermAnalysis plotTimeSeries

Examples
data(mouseData)
res = fitSSTimeSeries(obj=mouseData,feature="Actinobacteria",
                     class="status",id="mouseID",time="relativeTime",lvl='class',B=2)
**Usage**

```r
fitTimeSeries(
  obj,
  formula,
  feature,
  class,
  time,
  id,
  method = c("ssanova"),
  lvl = NULL,
  include = c("class", "time:class"),
  C = 0,
  B = 1000,
  norm = TRUE,
  log = TRUE,
  sl = 1000,
  featureOrder = NULL,
  ...
)
```

**Arguments**

- `obj`: metagenomeSeq MRexperiment-class object.
- `formula`: Formula for ssanova. Of the form: abundance ~ ... where ... includes any pData slot value.
- `feature`: Name or row of feature of interest.
- `class`: Name of column in phenoData of MRexperiment-class object for class membership.
- `time`: Name of column in phenoData of MRexperiment-class object for relative time.
- `id`: Name of column in phenoData of MRexperiment-class object for sample id.
- `method`: Method to estimate time intervals of differentially abundant bacteria (only ssanova method implemented currently).
- `lvl`: Vector or name of column in featureData of MRexperiment-class object for aggregating counts (if not OTU level).
- `include`: Parameters to include in prediction.
- `C`: Value for which difference function has to be larger or smaller than (default 0).
- `B`: Number of permutations to perform.
- `norm`: When aggregating counts to normalize or not.
- `log`: Log2 transform.
- `sl`: Scaling value.
- `featureOrder`: Hierarchy of levels in taxonomy as fData colnames
- `...`: Options for ssanova
**Value**

List of matrix of time point intervals of interest, Difference in abundance area and p-value, fit, area permutations, and call.

A list of objects including:

- `timeIntervals` - Matrix of time point intervals of interest, area of differential abundance, and pvalue.
- `data` - Data frame of abundance, class indicator, time, and id input.
- `fit` - Data frame of fitted values of the difference in abundance, standard error estimates and timepoints interpolated over.
- `perm` - Differential abundance area estimates for each permutation.
- `call` - Function call.

**See Also**

cumNorm fitSSTimeSeries plotTimeSeries

**Examples**

data(mouseData)
res = fitTimeSeries(obj=mouseData, feature="Actinobacteria", class="status", id="mouseID", time="relativeTime", lvl="Var", B=2)
Value

A list of objects including:

- logFC - the log fold-change estimates
- adjFactor - the adjustment factor based on the zero component
- se - standard error estimates
- fitln - parameters from the log-normal fit
- fitzero - parameters from the logistic fit
- zeroRidge - output from the ridge regression
- posRidge - output from the ridge regression
- tauPos - estimated tau^2 for positive component
- tauZero - estimated tau^2 for zero component
- exclude - features to exclude for various reasons, e.g. all zeros
- zeroExclude - features to exclude for various reasons, e.g. all zeros

See Also

cumNorm fitFeatureModel

---

**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**

```r
fitZig(
  obj,
  mod,
  zeroMod = NULL,
  useCSSoffset = TRUE,
  control = zigControl(),
  useMixedModel = FALSE,
  ...
)
```
fitZig

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.
useCSSoffset  Boolean, whether to include the default scaling parameters in the model or not.
control  The settings for fitZig.
useMixedModel  Estimate the correlation between duplicate features or replicates using duplicateCorrelation.
...  Additional parameters for duplicateCorrelation.

Value

A list of objects including:

- call - the call made to fitZig
- fit - 'MLArrayLM' Limma object of the weighted fit
- countResiduals - standardized residuals of the fit
- z - matrix of the posterior probabilities
- eb - output of eBayes, moderated t-statistics, moderated F-statistics, etc
- taxa - vector of the taxa names
- counts - the original count matrix input
- zeroMod - the zero model matrix
- zeroCoef - the zero model fitted results
- stillActive - convergence
- stillActiveNLL - nll at convergence
- dupcor - correlation of duplicates

See Also
cumNorm zigControl

Examples

# This is a simple demonstration
data(lungData)
k = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[, -k]
k = which(rowSums(MRcounts(lungTrim) > 0) < 30)
lungTrim = cumNorm(lungTrim)
lungTrim = lungTrim[-k,]
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~ smokingStatus)
# The maxit is not meant to be 1 - this is for demonstration/speed
settings = zigControl(maxit = 1, verbose = FALSE)
fit = fitZig(obj = lungTrim, mod = mod, control = settings)
fitZigResults-class

Class “fitZigResults” – a formal class for storing results from a fitZig call

Description
This class contains all of the same information expected from a fitZig call, but it is defined in the S4 style as opposed to being stored as a list.

Slots
- call: the call made to fitZig
- fit: 'MLArrayLM' Limma object of the weighted fit
- countResiduals: standardized residuals of the fit
- z: matrix of the posterior probabilities. It is defined as $z_{ij} = \text{pr}(\delta_{ij}=1 | \text{data})$
- zUsed: used in getZ
- eb: output of eBayes, moderated t-statistics, moderated F-statistics, etc
- taxa: vector of the taxa names
- counts: the original count matrix input
- zeroMod: the zero model matrix
- zeroCoef: the zero model fitted results
- stillActive: convergence
- stillActiveNLL: nll at convergence
- dupcor: correlation of duplicates

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 $\delta_{ij}$ = 1 if $y_{ij}$ is generated from the zero point mass as latent indicator variables. The density is defined as $f_{\text{zig}}(y_{ij} = \pi_{ij}(S_{ij}) \cdot f_0(y_{ij}) + (1 - \pi_{ij}(S_{ij})) f_{\text{count}}(y_{ij}; \mu_{ij}, \sigma_{ij}^2)$. The log-likelihood in this extended model is $(1 - \delta_{ij}) \log f_{\text{count}}(y; \mu_{ij}, \sigma_{ij}^2) + \delta_{ij} \log \pi_{ij}(S_{ij}) + (1 - \delta_{ij}) \log (1 - \pi_{ij}(S_{ij}))$. The responsibilities are defined as $z_{ij} = \text{pr}(\delta_{ij}=1 | \text{data})$.

Usage
getCountDensity(residuals, log = FALSE)
getEpsilon

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

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, nll0ld)

Arguments

nll Vector of size \( M \) with the current negative log-likelihoods.
nll0ld 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 $z_{ij} = \text{pr}(\delta_{ij} = 1 | \text{data and current values})$.

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)
getZ

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

description

esps Vector of size M (features) representing the relative difference between the new nll and old nll.
description
tol The threshold tolerance for the difference
description
stillActive A vector of size M booleans saying if a feature is still active or not.
description
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

Access 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(object)
Arguments

object  a MRexperiment object

Value

Library sizes

Author(s)

Joseph N. Paulson

Examples

data(lungData)
head(libSize(lungData))

---

libSize<-  \textit{Replace the library sizes in a MRexperiment object}

Description

Function to replace the scaling factors, aka the library sizes, of samples in a MRexperiment object.

Usage

\begin{verbatim}
## S4 replacement method for signature 'MRexperiment,numeric'
libSize(object) <- value
\end{verbatim}

Arguments

object  a MRexperiment object
value  vector of library sizes

Value

vector library sizes

Author(s)

Joseph N. Paulson

Examples

data(lungData)
head(libSize(lungData)<- rnorm(1))
loadBiom

Load objects organized in the Biom format.

Description
Wrapper to load Biom formatted object.

Usage
loadBiom(file)

Arguments
file The biom object filepath.

Value
A MRexperiment object.

See Also
loadMeta loadPhenoData newMRexperiment biom2MRexperiment

Examples
#library(biomformat)
rich_dense_file = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
x = loadBiom(rich_dense_file)
x

loadMeta

Load a count dataset associated with a study.

Description
Load a matrix of OTUs in a tab delimited format

Usage
loadMeta(file, sep = "\t")

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

Value

A list with objects 'counts' and 'taxa'.

See Also

`loadPhenoData`

Examples

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

---

**Description**

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

**Usage**

`loadMetaQ(file)`

**Arguments**

- `file`  
  Path and filename of the actual data file.

**Value**

An list with 'counts' containing the count data, 'taxa' containing the otu annotation, and 'otus'.

See Also

`loadMeta` `loadPhenoData`

Examples

```r
# see vignette
```
loadPhenoData  

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

**Description**

Load a matrix of metadata associated with a study.

**Usage**

```r
loadPhenoData(file, tran = TRUE, 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**

`loadMeta`

**Examples**

```r
dataDirectory <- system.file("extdata", package="metagenomeSeq")
clin = loadPhenoData(file.path(dataDirectory,"CHK_clinical.csv"),tran=TRUE)
```

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.

**Format**

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

Value

MRexperiment-class object of 16S lung samples.

References


---

**makeLabels**

*Function to make labels simpler*

---

**Description**

Beginning to transition to better axes for plots

**Usage**

```r
makeLabels(x = "samples", y = "abundance", norm, log)
```

**Arguments**

- `x` string for the x-axis
- `y` string for the y-axis
- `norm` is the data normalized?
- `log` is the data logged?

**Value**

vector of x,y labels

**Examples**

```r
metagenomeSeq::makeLabels(norm=TRUE, log=TRUE)
```
mergeMRexperiments  

Merge two MRexperiment objects together

Description

This function will take two MRexperiment objects and merge them together finding common OTUs. If there are OTUs not found in one of the two MRexperiments then a message will announce this and values will be coerced to zero for the second table.

Usage

mergeMRexperiments(x, y)

Arguments

x  
MRexperiment-class object 1.

y  
MRexperiment-class object 2.

Value

Merged MRexperiment-class object.

Examples

data(mouseData)
newobj = mergeMRexperiments(mouseData,mouseData)
newobj

# let me know if people are interested in an option to merge by keys instead of row names.
data(lungData)
newobj = mergeMRexperiments(mouseData,lungData)
newobj

mergeTable  

Merge two tables

Description

Merge two tables

Usage

mergeTable(x, y)
**Arguments**

- **x**
  
  Table 1.

- **y**
  
  Table 2.

**Value**

Merged table

---

**Description**

These functions may be removed completely in the next release.

**Usage**

`deprecated_metagenomeSeq_function(x, value, ...)`

**Arguments**

- **x**
  
  For assignment operators, the object that will undergo a replacement (object inside parenthesis).

- **value**
  
  For assignment operators, the value to replace with (the right side of the assignment).

- **...**
  
  For functions other than assignment operators, parameters to be passed to the modern version of the function (see table).

---

**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.

**Format**

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

**Value**

MRexperiment-class object of 16S mouse samples.
MRcoefs

Table of top-ranked features from fitZig or fitFeatureModel

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,
  adjustMethod = "fdr",
  alpha = 0.1,
  group = 0,
  eff = 0,
  numberEff = FALSE,
  counts = 0,
  file = NULL
)

Arguments

obj Output of fitFeatureModel or 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.
adjustMethod 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. Additionally, options using independent hypothesis weighting (IHW) are available. See MRihw for more details.
alpha Value for p-value significance threshold when running IHW. The default is set to 0.1

References

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

One of five choices, 0,1,2,3,4. 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. 4: no sorting.

eff

Filter features to have at least a "eff" quantile or number of effective samples.

numberEff

Boolean, whether eff should represent quantile (default/FALSE) or number.

counts

Filter features to have at least 'counts' counts.

file

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 fitFeatureModel MRtable MRfulltable

Examples

data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[, -k]
lungTrim = filterData(lungTrim, present = 30)
lungTrim = cumNorm(lungTrim, p = 0.5)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~ smokingStatus)
fit = fitZig(obj = lungTrim, mod = mod)
head(MRcoefs(fit))
###
fit = fitFeatureModel(obj = lungTrim, mod = mod)
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 log2 transform scale.
- **sl**: The value to scale by (default=1000).

### Value

Normalized or raw counts

### Author(s)

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

### Examples

```r
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.

```r
[  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 `posteriorProbs`.

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 (`posteriorProbs`))

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

### MRexperiment2biom

**MRexperiment to biom objects**

**Description**

Wrapper to convert MRexperiment objects to biom objects.

**Usage**

```r
MRexperiment2biom(
  obj,
  id = NULL,
  norm = FALSE,
  log = FALSE,
  sl = 1000,
  qiimeVersion = TRUE
)
```
Arguments

obj  The MRexperiment object.
id   Optional id for the biom matrix.
norm normalize count table
loglog2 transform count table
sl  scaling factor for normalized counts.
qiimeVersion Format fData according to QIIME specifications (assumes only taxonomy in fData).

Value

A biom object.

See Also

loadMeta loadPhenoData newMRexperiment loadBiom biom2MRexperiment

Usage

MRfulltable(
  obj,
  by = 2,
  coef = NULL,
  number = 10,
  taxa = obj@taxa,
  uniqueNames = FALSE,
  adjustMethod = "fdr",
  group = 0,
  eff = 0,
  numberEff = FALSE,
  ncounts = 0,
  file = NULL
)
Arguments

- **obj**  
  Output of fitFeatureModel or 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.

- **adjustMethod**  
  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 five choices: 0,1,2,3,4. 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. 4: no sorting.

- **eff**  
  Filter features to have at least a "eff" quantile or number of effective samples.

- **numberEff**  
  Boolean, whether eff should represent quantile (default/TRUE) or number.

- **ncounts**  
  Filter features to those with at least 'counts' counts.

- **file**  
  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  
- fitFeatureModel  
- MRcoefs  
- MRtable  
- fitPA

Examples

```r
data(lungData)
k = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[-k]
lungTrim = filterData(lungTrim, present=30)
lungTrim = cumNorm(lungTrim, p=0.5)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
fit = fitZig(obj = lungTrim, mod=mod)
head(MRfulltable(fit))
####
fit = fitFeatureModel(obj = lungTrim, mod=mod)
head(MRfulltable(fit))
```
MRihw runs IHW within a MRcoefs() call

Description
Function used in MRcoefs() when "IHW" is set as the p value adjustment method

Usage
MRihw(obj, ...)

Arguments
obj Either a fitFeatureModelResults or fitZigResults object
... other parameters

p a vector of pvalues extracted from obj
adjustMethod Value specifying which adjustment method and which covariate to use for IHW pvalue adjustment. For obj of class fitFeatureModelResults-class, options are "ihw-abundance" (median feature count per row) and "ihw-ubiquity" (number of non-zero features per row). For obj of class fitZigResults-class, options are "ihw-abundance" (weighted mean per feature) and "ihw-ubiquity" (number of non-zero features per row).
alpha pvalue significance level specified for IHW call. Default is 0.1
MRihw, fitZigResults-method

MRihw runs IHW within a MRcoefs() call

Description

Function used in MRcoefs() when "IHW" is set as the p value adjustment method.

Usage

## S4 method for signature 'fitZigResults'
MRihw(obj, p, adjustMethod, alpha)

Arguments

- **obj**: Either a fitFeatureModelResults or fitZigResults object
- **p**: a vector of pvalues extracted from obj
- **adjustMethod**: Value specifying which adjustment method and which covariate to use for IHW pvalue adjustment. For obj of class fitFeatureModelResults-class, options are "ihw-abundance" (median feature count per row) and "ihw-ubiquity" (number of non-zero features per row). For obj of class fitZigResults-class, options are "ihw-abundance" (weighted mean per feature) and "ihw-ubiquity" (number of non-zero features per row).
- **alpha**: pvalue significance level specified for IHW call. Default is 0.1

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,
adjustMethod = "fdr",
group = 0,
eff = 0,
numberEff = FALSE,
ncounts = 0,
file = NULL
)

Arguments

obj Output of fitFeatureModel or 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.
adjustMethod 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 five choices, 0,1,2,3,4. 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. 4: no sorting.
eff Filter features to have at least a "eff" quantile or number of effective samples.
numberEff Boolean, whether eff should represent quantile (default/FALSE) or number.
ncounts Filter features to have at least 'counts' of counts.
file Name of file, including location, to save the table.

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

See Also

fitZig fitFeatureModel MRcoefs MRfulltable

Examples

data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[,,-k]
lungTrim=filterData(lungTrim,present=30)
lungTrim=cumNorm(lungTrim,p=0.5)
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
fit = fitZig(obj = lungTrim,mod=mod)
head(MRtable(fit))
####
fit = fitFeatureModel(obj = lungTrim,mod=mod)
head(MRtable(fit))

---

**newMReperiment**  
*Create a MReperiment object*

**Description**

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

**Usage**

```r
newMReperiment(
  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 **MReperiment-class** and eSet (from the Biobase package) for the meaning of the various slots.

**Value**

an object of class MReperiment
**Author(s)**

Joseph N Paulson

**Examples**

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

---

<table>
<thead>
<tr>
<th>normFactors</th>
<th>Access the normalization factors in a MRexperiment object</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```r
normFactors(object)
```

**Arguments**

- `object`: a MRexperiment object

**Value**

Normalization scaling factors

**Author(s)**

Joseph N. Paulson

**Examples**

```r
data(lungData)
head(normFactors(lungData))
```
normFactors <- Replace the normalization factors in a MRexperiment object

Description

Function to replace the scaling factors, aka the normalization factors, of samples in a MRexperiment object.

Usage

```r
## S4 replacement method for signature 'MRexperiment,numeric'
normFactors(object) <- value
```

Arguments

- `object` a MRexperiment object
- `value` vector of normalization scaling factors

Value

Normalization scaling factors

Author(s)

Joseph N. Paulson

Examples

```r
data(lungData)
head(normFactors(lungData) <- rnorm(1))
```

plotBubble Basic plot of binned vectors.

Description

This function plots takes two vectors, calculates the contingency table and plots circles sized by the contingency table value. Optional significance vectors of the values significant will shade the circles by proportion of significance.
plotBubble

Usage

plotBubble(
  yvector,
  xvector,
  sigvector = NULL,
  nbreaks = 10,
  ybreak = quantile(yvector, p = seq(0, 1, length.out = nbreaks)),
  xbreak = quantile(xvector, p = seq(0, 1, length.out = nbreaks)),
  scale = 1,
  local = FALSE,
  ...
)

Arguments

yvector  A vector of values represented along y-axis.
xvector  A vector of values represented along x-axis.
sigvector  A vector of the names of significant features (names should match x/yvector).
nbreaks  Number of bins to break yvector and xvector into.
ybreak  The values to break the yvector at.
xbreak  The values to break the xvector at.
scale  Scaling of circle bin sizes.
local  Boolean to shade by significant bin numbers (TRUE) or overall proportion (FALSE).
...
  Additional plot arguments.

Value

A matrix of features along rows, and the group membership along columns.

See Also

plotMRheatmap

Examples

data(mouseData)
mouseData = mouseData[which(rowSums(mouseData)>139),]
sparsity = rowMeans(MRcounts(mouseData)==0)
lor = log(fitPA(mouseData,cl=pData(mouseData)[,3])$oddsRatio)
plotBubble(lor,sparsity,main="lor ~ sparsity")
# Example 2
x = runif(100000)
y = runif(100000)
plotBubble(y,x)
plotClassTimeSeries  Plot abundances by class

Description
Plot the abundance of values for each class using a spline approach on the estimated full model.

Usage
plotClassTimeSeries(
  res,
  formula,
  xlab = "Time",
  ylab = "Abundance",
  color0 = "black",
  color1 = "red",
  include = c("1", "class", "time:class"),
  ...
)

Arguments
- res: Output of fitTimeSeries function
- formula: Formula for ssanova. Of the form: abundance ~ ... where ... includes any pData slot value.
- xlab: X-label.
- ylab: Y-label.
- color0: Color of samples from first group.
- color1: Color of samples from second group.
- include: Parameters to include in prediction.
- ...: Extra plotting arguments.

Value
Plot for abundances of each class using a spline approach on estimated null model.

See Also
fitTimeSeries

Examples
```
data(mouseData)
res = fitTimeSeries(obj=mouseData, feature="Actinobacteria",
  class="status", id="mouseID", time="relativeTime", lvl='class', B=10)
plotClassTimeSeries(res, pch=21, bg=res$data$class, ylim=c(0,8))
```
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, norm = TRUE, log = TRUE, fun = cor, ...)

Arguments

obj A MRexperiment object with count data.

n The number of features to plot. This chooses the "n" features with greatest variance.

norm Whether or not to normalize the counts - if MRexperiment object.

log Whether or not to log2 transform the counts - if MRexperiment object.

fun Function to calculate pair-wise relationships. Default is pearson correlation

... Additional plot arguments.

Value

plotted correlation matrix

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))
plotFeature  Basic plot function of the raw or normalized data.

Description
This function plots the abundance of a particular OTU by class. The function is the typical manhattan plot of the abundances.

Usage
plotFeature(
  obj,
  otuIndex,
  classIndex,
  col = "black",
  sort = TRUE,
  sortby = NULL,
  norm = TRUE,
  log = TRUE,
  sl = 1000,
  ...
)

Arguments
  obj        A MRexperiment object with count data.
  otuIndex   The row to plot
  classIndex A list of the samples in their respective groups.
  col        A vector to color samples by.
  sort       Boolean, sort or not.
  sortby     Default is sort by library size, alternative vector for sorting
  norm       Whether or not to normalize the counts - if MRexperiment object.
  log        Whether or not to log2 transform the counts - if MRexperiment object.
  sl         Scaling factor - if MRexperiment and norm=TRUE.
  ...        Additional plot arguments.

Value
  counts and classindex

See Also
cumNorm
Examples

```r
data(mouseData)
classIndex=list(Western=which(pData(mouseData)$diet=="Western"))
classIndex$BK=which(pData(mouseData)$diet=="BK")
etuIndex=8770

par(mfrow=c(2,1))
dates=pData(mouseData)$date
plotFeature(mouseData,norm=FALSE,log=FALSE,otuIndex,classIndex,
col=dates,sortby=dates,ylab="Raw reads")
```

---

**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

```r
plotGenus(
  obj,
  otuIndex,
  classIndex,
  norm = TRUE,
  log = TRUE,
  no = 1:length(otuIndex),
  labs = TRUE,
  xlab = NULL,
  ylab = NULL,
  jitter = TRUE,
  jitter.factor = 1,
  pch = 21,
  ...
)
```

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 - if MRexperiment object.
- `log` Whether or not to log2 transform the counts - if MRexperiment object.
- `no` Which of the otuIndex to plot.
plotMRheatmap

Description

This function plots a heatmap of the 'n' features with greatest variance across rows (or other statistic).

Usage

plotMRheatmap(obj, n, norm = TRUE, log = TRUE, fun = sd, ...)

Arguments

obj

A MRexperiment object with count data.

n

The number of features to plot. This chooses the 'n' features of greatest positive statistic.

norm

Whether or not to normalize the counts - if MRexperiment object.

log

Whether or not to log2 transform the counts - if MRexperiment object.

fun

Function to select top 'n' features.

... Additional plot arguments.

Value

plotted data

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)$family)

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)")
Value

plotted matrix

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)
#### version using sd
plotMRheatmap(obj=mouseData,n=200,cexRow = 0.4,cexCol = 0.4,trace="none",
              col = heatmapCols,ColSideColors = heatmapColColors)
#### version using MAD
plotMRheatmap(obj=mouseData,n=50,fun=mad,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 = TRUE,
    comp = 1:2,
    norm = TRUE,
    log = TRUE,
    usePCA = TRUE,
    useDist = FALSE,
    distfun = stats::dist,
    dist.method = "euclidian",
    n = NULL,
    ...
)
plotOTU

Arguments

obj: A MRexperiment object or count matrix.
tran: Transpose the matrix.
comp: Which components to display
norm: Whether or not to normalize the counts - if MRexperiment object.
log: Whether or not to log2 the counts - if MRexperiment object.
usePCA: TRUE/FALSE whether to use PCA or MDS coordinates (TRUE is PCA).
useDist: TRUE/FALSE whether to calculate distances.
distfun: Distance function, default is stats::dist
dist.method: If useDist==TRUE, what method to calculate distances.
n: Number of features to make use of in calculating your distances.
...: Additional plot arguments.

Value

coordinates

See Also

cumNormMat

Examples

data(mouseData)
c1 = pData(mouseData)[,3]
plotOrd(mouseData,tran=TRUE,useDist=TRUE,pch=21,bg=factor(c1),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,
...
)

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 log2 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.
...           Additional plot arguments.

Value

Plotted values

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, ...)

Arguments

obj  
A MRexperiment object with count data or matrix.

cl  
Vector of classes for various samples.

...  
Additional plot arguments.

Value

Library size and number of detected features

See Also

plotOrd, plotMRheatmap, plotCorr, plotOTU, plotGenus

Examples

data(mouseData)
cl = factor(pData(mouseData)[,3])
res = plotRare(mouseData, cl=cl, 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)
plotTimeSeries

Plot difference function for particular bacteria

Description

Plot the difference in abundance for significant features.

Usage

```r
plotTimeSeries(
  res,
  C = 0,
  xlab = "Time",
  ylab = "Difference in abundance",
  main = "SS difference function prediction",
  ...
)
```

Arguments

- `res` Output of `fitTimeSeries` function
- `C` Value for which difference function has to be larger or smaller than (default 0).
- `xlab` X-label.
- `ylab` Y-label.
- `main` Main label.
- `...` Extra plotting arguments.

Value

Plot of difference in abundance for significant features.

See Also

`fitTimeSeries`

Examples

```r
data(mouseData)
res = fitTimeSeries(obj=mouseData,feature="Actinobacteria",
  class="status",id="mouseID",time="relativeTime",lvl='class',B=10)
plotTimeSeries(res)
```
posteriorProbs

Access the posterior probabilities that results from analysis

Description

Accessing the posterior probabilities following a run through fitZig

Usage

posteriorProbs(obj)

Arguments

obj

a MRexperiment object.

Value

Matrix of posterior probabilities

Author(s)

Joseph N. Paulson

Examples

# This is a simple demonstration
data(lungData)
k = grep("Extraction.Control",pData(lungData)$SampleType)
lungTrim = lungData[,-k]
k = which(rowSums(MRcounts(lungTrim)>0)<30)
lungTrim = cumNorm(lungTrim)
lungTrim = lungTrim[-k,]
smokingStatus = pData(lungTrim)$SmokingStatus
mod = model.matrix(~smokingStatus)
# The maxit is not meant to be 1 -- this is for demonstration/speed
settings = zigControl(maxit=1,verbose=FALSE)
fit = fitZig(obj = lungTrim,mod=mod,control=settings)
head(posteriorProbs(lungTrim))
**returnAppropriateObj**  
*Check if MRexperiment or matrix and return matrix*

**Description**
Function to check if object is a MRexperiment class or matrix

**Usage**

```r
returnAppropriateObj(obj, norm, log, sl = 1000)
```

**Arguments**
- `obj`: a MRexperiment or matrix object
- `norm`: return a normalized MRexperiment matrix
- `log`: return a log transformed MRexperiment matrix
- `sl`: scaling value

**Value**
Matrix

**Examples**

```r
data(lungData)
head(returnAppropriateObj(lungData,norm=FALSE,log=FALSE))
```

---

**ssFit**  
*smoothing-splines anova fit*

**Description**
Sets up a data-frame with the feature abundance, class information, time points, sample ids and returns the fitted values for the fitted model.

**Usage**

```r
ssFit(
    formula,
    abundance,
    class,
    time,
    id,
    include = c("class", "time:class"),
    pd,
    ...
)
```
Arguments

formula Formula for ssanova. Of the form: abundance ~ ... where ... includes any pData slot value.
abundance Numeric vector of abundances.
class Class membership (factor of group membership).
time Time point vector of relative times (same length as abundance).
id Sample / patient id.
include Parameters to include in prediction.
pd Extra variable.
... Extra parameters for ssanova function (see ?ssanova).

Value

A list containing:

- data : Inputed data
- fit : The interpolated / fitted values for timePoints
- se : The standard error for CI intervals
- timePoints : The time points interpolated over

See Also

cumNorm fitTimeSeries ssPermAnalysis ssPerm ssIntervalCandidate

Examples

# Not run

ssIntervalCandidate calculate interesting time intervals

Description

Calculates time intervals of interest using SS-Anova fitted confidence intervals.

Usage

ssIntervalCandidate(fit, standardError, timePoints, positive = TRUE, C = 0)
Arguments

**fit**  SS-Anova fits.
**standardError**  SS-Anova se estimates.
**timePoints**  Time points interpolated over.
**positive**  Positive region or negative region (difference in abundance is positive/negative).
**C**  Value for which difference function has to be larger or smaller than (default 0).

Value

Matrix of time point intervals of interest

See Also

`cumNorm` `fitTimeSeries` `ssFit` `ssPerm` `ssPermAnalysis`

Examples

```r
# Not run
```

---

**ssPerm**

class permutations for smoothing-spline time series analysis

Description

Creates a list of permuted class memberships for the time series permutation tests.

Usage

```r
ssPerm(df, B)
```

Arguments

**df**  Data frame containing class membership and sample/patient id label.
**B**  Number of permutations.

Value

A list of permuted class memberships

See Also

`cumNorm` `fitTimeSeries` `ssFit` `ssPermAnalysis` `ssIntervalCandidate`

Examples

```r
# Not run
```
ssPermAnalysis

smooth-splines anova fits for each permutation

Description

Calculates the fit for each permutation and estimates the area under the null (permuted) model for interesting time intervals of differential abundance.

Usage

ssPermAnalysis(
  data,
  formula,
  permList,
  intTimes,
  timePoints,
  include = c("class", "time:class"),
  ...
)

Arguments

data      Data used in estimation.
formula   Formula for ssanova. Of the form: abundance ~ ... where ... includes any pData slot value.
permList  A list of permuted class memberships
intTimes  Interesting time intervals.
timePoints Time points to interpolate over.
include   Parameters to include in prediction.
...       Options for ssanova

Value

A matrix of permuted area estimates for time intervals of interest.

See Also

cumNorm fitTimeSeries ssFit ssPerm ssIntervalCandidate

Examples

# Not run
trapz  
Trapezoidal Integration

Description

Compute the area of a function with values 'y' at the points 'x'. Function comes from the pracma package.

Usage

\[
\text{trapz}(x, y)
\]

Arguments

- \(x\): x-coordinates of points on the x-axis
- \(y\): y-coordinates of function values

Value

Approximated integral of the function from 'min(x)' to 'max(x)'. Or a matrix of the same size as 'y'.

Examples

# Calculate the area under the sine curve from 0 to pi:
\[
n \leftarrow 101
\]
\[
x \leftarrow \text{seq}(0, \pi, \text{len} = n)
\]
\[
y \leftarrow \sin(x)
\]
\[
\text{trapz}(x, y) \quad \Rightarrow 1.999835504
\]

# Use a correction term at the boundary: \(-h^2/12*(f'(b)-f'(a))\)
\[
h \leftarrow x[2] - x[1]
\]
\[
ca \leftarrow (y[2]-y[1]) / h
\]
\[
cb \leftarrow (y[n]-y[n-1]) / h
\]
\[
\text{trapz}(x, y) - h^2/12 * (cb - ca) \Rightarrow 1.999999969
\]

Description

With a list of fitTimeSeries results, generate an MRexperiment that can be plotted with metavizr

ts2MRexperiment  
With a list of fitTimeSeries results, generate an MRexperiment that can be plotted with metavizr

Description

With a list of fitTimeSeries results, generate an MRexperiment that can be plotted with metavizr
Usage

```r
ts2MReperiment(
  obj,
  sampleNames = NULL,
  sampleDescription = "timepoints",
  taxonomyLevels = NULL,
  taxonomyHierarchyRoot = "bacteria",
  taxonomyDescription = "taxonomy",
  featuresOfInterest = NULL,
  featureDataOfInterest = NULL
)
```

Arguments

- **obj**: Output of `fitMultipleTimeSeries`
- **sampleNames**: Sample names for plot
- **sampleDescription**: Description of samples for plot axis label
- **taxonomyLevels**: Feature names for plot
- **taxonomyHierarchyRoot**: Root of feature hierarchy for MReperiment
- **taxonomyDescription**: Description of features for plot axis label
- **featuresOfInterest**: The features to select from the `fitMultipleTimeSeries` output
- **featureDataOfInterest**: Feature data for the resulting MReperiment

Value

MReperiment that contains `fitTimeSeries` data, featureData, and phenoData

See Also

- `fitTimeSeries`
- `fitMultipleTimeSeries`

Examples

```r
data(mouseData)
res = fitMultipleTimeSeries(obj=mouseData,lvl='phylum',class="status",
                            id="mouseID",time="relativeTime",B=1)
obj = ts2MReperiment(res)
obj
```
### uniqueFeatures

**Table of features unique to a group**

#### Description

Creates a table of features, their index, number of positive samples in a group, and the number of reads in a group. Can threshold features by a minimum no. of reads or no. of samples.

#### Usage

```r
uniqueFeatures(obj, cl, nsamples = 0, nreads = 0)
```

#### Arguments

- `obj`: Either a MRexperiment object or matrix.
- `cl`: A vector representing assigning samples to a group.
- `nsamples`: The minimum number of positive samples.
- `nreads`: The minimum number of raw reads.

#### Value

Table of features unique to a group

#### Examples

```r
data(mouseData)
head(uniqueFeatures(mouseData[1:100,], cl = pData(mouseData)[,3]))
```

### wrenchNorm

**Computes normalization factors using wrench instead of cumNorm**

#### Description

Calculates normalization factors using method published by M. Sentil Kumar et al. (2018) to compute normalization factors which considers compositional bias introduced by sequencers.

#### Usage

```r
wrenchNorm(obj, condition)
```

#### Arguments

- `obj`: an MRexperiment object
- `condition`: case control label that wrench uses to calculate normalization factors
Value

an MRexperiment object with updated normalization factors. Accessible by \texttt{normFactors}.

See Also

cumNorm fitZig

Examples

data(mouseData)
mouseData <- wrenchNorm(mouseData, condition = mouseData$diet)
head(normFactors(mouseData))

---

\texttt{zigControl} \hspace{1cm} \textit{Settings for the fitZig function}

Description

Settings for the fitZig function

Usage

\texttt{zigControl(}
  \texttt{tol = 1e-04, maxit = 10, verbose = TRUE, dfMethod = \"modified\", pvalMethod = \"default\")}

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.
dfMethod Either ‘default’ or ‘modified’ (by responsibilities).
pvalMethod Either ‘default’ or ‘bootstrap’.

Value

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

Note

\texttt{fitZig} makes use of \texttt{zigControl}. 
See Also

\textit{fitZig cumNorm plotOTU}

Examples

\begin{verbatim}
control = zigControl(tol=1e-10,maxit=10,verbose=FALSE)
\end{verbatim}
Index

* package
  metagenomeSeq-package, 4
  [,MReperiment,ANY,ANY,ANY-method
   (MReperiment), 48
  [,MReperiment-method
   (MReperiment), 48
aggregateBySample, 4
aggregateByTaxonomy, 5
aggSamp (aggregateBySample), 4
aggTax (aggregateByTaxonomy), 5
biom2MRexperiment, 6, 40, 50
calcNormFactors, 7
calcPosComponent, 8
calcShrinkParameters, 8
calcStandardError, 9
calculateEffectiveSamples, 9
calcZeroAdjustment, 10
calcZeroComponent, 10
colMeans, MReperiment-method
   (MReperiment), 48
colSums, MReperiment-method
   (MReperiment), 48
correctIndices, 11, 12
correlationTest, 11, 12
corTest (correlationTest), 12
cumNorm, 7, 13, 14–16, 19, 20, 23, 24, 26, 27, 29, 31–33, 49, 61, 63, 66, 71–73, 77, 78
cumNormMat, 14, 49, 60, 64, 65
cumNormStat, 13, 14, 16, 49
cumNormStatFast, 7, 15, 15
deprecated_metagenomeSeq_function
   (metagenomeSeq-deprecated), 45
doCountMStep, 16
doEStep, 17
doZeroMStep, 18
exportMat, 18
exportMatrix (exportMat), 18
exportStats, 19
expSummary, 20
expSummary, MReperiment-method
   (expSummary), 20
extractMR, 21
filterData, 22
fitDO, 22, 27
fitFeatureModel, 8–10, 23, 32, 47, 51, 54
fitFeatureModelResults-class, 24
fitLogNormal, 25
fitMeta, 23, 27
fitLogNormal, 26, 75
fitPA, 23, 27, 51
fitSSTimeSeries, 26, 28, 31
fitTimeSeries, 26, 29, 59, 68, 71–73, 75
fitZeroLogNormal, 8–10, 31
fitZig, 7, 10, 13–18, 23, 27, 32, 35–38, 47, 51, 54, 69, 77, 78
fitZigResults-class, 34
genusPlot (plotGenus), 62
getCountDensity, 34
getEpsilon, 35
getNegativeLogLikelihoods, 36
geti, 36
getZ, 34, 37
isItStillActive, 38
libSize, 38, 49
libSize, MReperiment-method
   (MReperiment), 48
libSize<-, 39
libSize<-, MReperiment, numeric-method
   (libSize<-), 39
load_biom (metagenomeSeq-deprecated), 45
load_meta (metagenomeSeq-deprecated), 45
load_metaQ (metagenomeSeq-deprecated), 45
load_phenoData
  (metagenomeSeq-deprecated), 45
loadBiom, 7, 40, 50
loadMeta, 7, 40, 41, 42, 50
loadMetaQ, 41
loadPhenoData, 7, 40, 41, 42, 50
lungData, 42

makeLabels, 43
mergeMRexperiments, 44
mergeTable, 44
metagenomeSeq (metagenomeSeq-package), 4
metagenomeSeq-deprecated, 45
metagenomicLoader (loadMeta), 40
mouseData, 45
MRcoefs, 10, 46, 51, 54
MRcounts, 47, 49
MRcounts, MRexperiment-method
  (MRcounts), 47
MRexperiment, 48
MRexperiment-class (MRexperiment), 48
MRexperiment2biom, 49
MRfulltable, 10, 47, 50, 54
MRihw, 46, 52
MRihw, fitFeatureModelResults-method, 52
MRihw, fitZigResults-method, 53
MRtable, 47, 51, 53

newMRexperiment, 7, 40, 48–50, 55
normFactors, 49, 56, 77
normFactors, MRexperiment-method
  (MRexperiment), 48
normFactors<-, 57
normFactors<-, MRexperiment, numeric-method
  (normFactors<->), 57

p.adjust, 23, 27, 46, 51, 54
phenoData (loadPhenoData), 42
plotBubble, 57
plotClassTimeSeries, 59
plotCorr, 60, 67
plotFeature, 61
plotGenus, 62, 67
plotMRheatmap, 58, 63, 67
plotOrd, 64, 67

plotOTU, 65, 67, 78
plotRare, 67
plotTimeSeries, 29, 31, 68
posteriorProbs, 49, 69
posteriorProbs, MRexperiment-method
  (posteriorProbs), 69
qiimeLoader (loadMetaQ), 41
quantile, 20

returnAppropriateObj, 70
rowMeans, MRexperiment-method
  (MRexperiment), 48
rowSums, MRexperiment-method
  (MRexperiment), 48

settings2 (zigControl), 77
ssFit, 29, 70, 72, 73
ssIntervalCandidate, 29, 71, 72, 73
ssPerm, 29, 71, 72, 73
ssPermAnalysis, 29, 71, 72, 73

trapz, 74
ts2MRexperiment, 74

uniqueFeatures, 76
wrenchNorm, 76

zigControl, 33, 77