Package ‘DEqMS’

April 3, 2024

Version 1.20.0
Date 2019/11/09
Title a tool to perform statistical analysis of differential protein expression for quantitative proteomics data.
Author Yafeng Zhu
Maintainer Yafeng Zhu <yafeng.zhu@outlook.com>
Depends R(>= 3.5), graphics, stats, ggplot2, matrixStats, limma(>= 3.34)
Suggests BiocStyle, knitr, markdown, markdown,plyr, reshape2, farms, utils, ggrepel, ExperimentHub, LSD
LazyLoad yes
Description DEqMS is developed on top of Limma. However, Limma assumes same prior variance for all genes. In proteomics, the accuracy of protein abundance estimates varies by the number of peptides/PSMs quantified in both label-free and labelled data. Proteins quantification by multiple peptides or PSMs are more accurate. DEqMS package is able to estimate different prior variances for proteins quantified by different number of PSMs/peptides, therefore achieving better accuracy. The package can be applied to analyze both label-free and labelled proteomics data.
License LGPL
biocViews ImmunoOncology, Proteomics, MassSpectrometry, Preprocessing, DifferentialExpression, MultipleComparison, Normalization, Bayesian, ExperimentHubSoftware
VignetteBuilder knitr
BugReports https://github.com/yafeng/DEqMS/issues
git_url https://git.bioconductor.org/packages/DEqMS
git_branch RELEASE_3_18
git_last_commit 4050c40
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-04-03
equalMedianNormalization

normalize to have equal medians in all samples

Description
This function is to normalize out the differences of protein medians in different samples

Usage
equalMedianNormalization(dat)

Arguments
dat an numeric data frame or matrix containing protein relative abundance in log2 scale

Value
a data frame or matrix with normalized protein relative abundance

Author(s)
Yafeng Zhu

Examples
library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh["EH1663"]
dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])
# use the 3 ctrl samples as reference channels to calculate log2 ratio
dat.gene = medianSummary(dat.psm.log, group_col = 2, ref_col = c(3,7,10))
dat.gene.nm = equalMedianNormalization(dat.gene)

Description
This function is to calculate proteins’ relative abundance by factor analysis.

Usage
farmsSummary(dat, group_col = 2)

Arguments
- dat: an data frame with raw peptide or psm intensities.
- group_col: the column by which peptides/psm intensity are grouped. Usually it is the gene/protein id column. Default is 2.

Value
a data frame containing protein relative abundance estimate in log2 scale.

Author(s)
Yafeng Zhu

Examples
library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh["EH1663"]
# farms method does not tolerate missing values
dat.gene = farmsSummary(dat.psm, group_col = 2)
Description

This function is to calculate proteins' relative abundance by median method.

Usage

medianSummary(dat, group_col=2, ref_col)

Arguments

dat: an data frame with peptide/psm intensities in log2 scale

group_col: the column by which peptides/psm intensity are grouped. Usually the gene/protein id column. Default is 2

ref_col: an integer vector indicating the column(s) used as denominator to calculate relative petide ratio.

Value

a data frame containing protein relative abundance estimate in log2 scale

Author(s)

Yafeng Zhu

Examples

```r
library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh[["EH1663"]]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])
# use the 3 ctrl samples as reference channels to calculate log2 ratio
dat.gene = medianSummary(dat.psm.log, group_col = 2, ref_col = c(3,7,10))
```
medianSweeping

**medianSweeping**

*summarize peptide/PSM intensity into protein level relative abundance estimate by median sweeping method*

---

**Description**

This function is to calculate proteins’ relative abundance by median sweeping method

**Usage**

```r
medianSweeping(dat, group_col = 2)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td>an data frame with peptide/PSM intensities in log2 scale</td>
</tr>
<tr>
<td>group_col</td>
<td>the column by which peptides/PSM intensity are grouped. Usually the gene/protein id column. Default is 2</td>
</tr>
</tbody>
</table>

**Value**

a data frame with protein relative abundance estimate in log2 scale

**Author(s)**

Yafeng Zhu

**Examples**

```r
library(ExperimentHub)
eh = ExperimentHub(localHub = TRUE)
query(eh, "DEqMS")
dat.psm = eh[["EH1663"]]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log, group_col = 2)
```
medpolishSummary

summarize peptide/PSM intensity into protein level relative abundance estimate by Turkey median polish procedure

Description

This function is to calculate proteins’ relative abundance by Turkey median polish.

Usage

medpolishSummary(dat, group_col=2)

Arguments

dat      an data frame containing peptide/psm intensities in log2 scale

group_col the column by which peptides/psm intensity are grouped. Usually the gene/protein column. Default is 2

Value

a data frame containing protein relative abundance estimate in log2 scale

Author(s)

Yafeng Zhu

Examples

library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh[["EH1663"]]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene = medpolishSummary(dat.psm.log, group_col=2)
outputResult

output the DEqMS analysis results in a data frame

Description
This function is to generate DEqMS outputs in a data frame.

Usage

outputResult(fit, coef_col=1)

Arguments

fit
an list object produced by spectraCounteBayes function

coef_col
is an integer indicating the column of fit$coefficients for which corresponding t-statistics and p-values are extracted in the output

Value

a data frame object with the last three columns being:
sca.t - Peptide or Spectra Count Adjusted posterior t-value
sca.P.Value - Adjusted posterior p-value
sca.adj - sca.P.Value adjusted by BH method

Author(s)

Yafeng Zhu

Examples

library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh[['EH1663']]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log,group_col = 2)

psm.count.table = as.data.frame(table(dat.psm$gene)) # generate PSM count table
rownames(psm.count.table)=psm.count.table$Var1

cond = c("ctrl","miR191","miR372","miR519","ctrl","miR372","miR519","ctrl","miR191","miR372")

sampleTable <- data.frame(
  row.names = colnames(dat.psm)[3:12],
  cond = as.factor(cond)
  )
peptideProfilePlot

```r
gene.matrix = as.matrix(dat.gene.nm)
design = model.matrix(~cond,sampleTable)

fit1 <- eBayes(lmFit(gene.matrix,design))
# add PSM count for each gene
fit1$count <- psm.count.table[rownames(fit1$coefficients),2]

fit2 = spectraCounteBayes(fit1)

deqMS.results = outputResult(fit2, coef_col=3)
```

**Description**

This function is to plot log2 intensities of all peptides for one gene in different samples.

**Usage**

```r
peptideProfilePlot(dat, col=2, gene)
```

**Arguments**

- **dat**  
a data frame with peptide/psm log2 intensities
- **col**  
an integer indicates the column number where the gene protein id is. default is 2, assuming the gene/protein is in the second column
- **gene**  
an character indicates the gene name/id to be plotted

**Value**

return a ggplot2 object

**Author(s)**

Yafeng Zhu

**Examples**

```r
library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh[["EH1663"]]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

peptideProfilePlot(dat.psm.log,col=2,gene="TGFBR2")
```
Residualplot

plot the residuals against the number of quantified peptides/PSMs.

Description

This function is to plot the residuals of fit model on the vertical axis and the peptide or PSM count on the horizontal axis.

Usage

Residualplot(fit, xlab="log2(count)", ylab="Variance(fitted - observed)", main="")

Arguments

fit an object returned from spectraCounteBayes function
xlab the title for x axis
ylab the title for y axis
main the title for the figure

Value

return a plot graphic

Author(s)

Yafeng Zhu

Examples

library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh["EH1663"]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log, group_col = 2)

psm.count.table = as.data.frame(table(dat.psm$gene)) # generate PSM count table
rownames(psm.count.table)=psm.count.table$Var1

cond = c("ctrl","miR191","miR372","miR519","ctrl",
"miR372","miR519","ctrl","miR191","miR372")

sampleTable <- data.frame(
row.names = colnames(dat.psm)[3:12],
...
cond = as.factor(cond)

gene.matrix = as.matrix(dat.gene.nm)
design = model.matrix(~cond,sampleTable)

fit1 <- eBayes(lmFit(gene.matrix,design))
# add PSM count for each gene
fit1$count <- psm.count.table[rownames(fit1$coefficients),2]

fit2 = spectraCounteBayes(fit1)

Residualplot(fit2,xlab="log2(PSM count)",main="TMT data PXD004163")

---

spectraCounteBayes

Peptide/Spectra Count Based Empirical Bayes Statistics for Differential Expression

Description

This function is to calculate peptide/PSM count adjusted t-statistics, p-values.

Usage

spectraCounteBayes(fit, fit.method="loess", coef_col)

Arguments

- **fit**: an list object produced by Limma eBayes function, it should have one additional attribute $count, which stored the peptide or PSM count quantified for the gene in label-free or isobaric labelled data.
- **fit.method**: the method used to fit variance against the number of peptides/PSMs count quantified. Two available methods: "loess","nls" and "spline". default "loess"."loess" uses loess and span = 0.75, "nls" uses a explicit formula y=a+b/x. "spline" uses smooth.spline and "generalized cross-validation" for smoothing parameter computation. For "nls", independent variable x is peptide/PSM count, response y is pooled variance (fit$sigma^2). For "loess" and "spline" method, both x and y are log transformed before applying the two methods. In most of time, "loess" is sufficient. To quickly assess the fit model, use VarianceScatterplot and Residualplot functions.
- **coef_col**: an integer vector indicating the column(s) of fit$coefficients for which the function is to be performed. if not specified, all coefficients are used.

Details

This function adjusts the T-statistics and p-values for quantitative MS proteomics experiment according to the number of peptides/PSMs used for quantification. The method is similar in nature to intensity-based Bayes method (Maureen A. Sartor et al BMC Bioinformatics 2006).
Value

a list object with the following components

count  Peptide or PSM count used for quantification
sca.t  Spectra Count Adjusted posterior t-value
sca.p  Spectra Count Adjusted posterior p-value
sca.dfprior  Spectra Count Adjusted prior degrees of freedom
sca.priorvar  Spectra Count Adjusted prior variance
sca.postvar  Spectra Count Adjusted posterior variance
model  fitted model
fit.method  The method used to fit the model

Author(s)

Yafeng Zhu

Examples

library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh["EH1663"]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log,group_col = 2)

psm.count.table = as.data.frame(table(dat.psm$gene)) # generate PSM count table
rownames(psm.count.table)=psm.count.table$Var1

cond = c("ctrl","miR191","miR372","miR519","ctrl","miR372","miR519","ctrl","miR191","miR372")

sampleTable <- data.frame(
  row.names = colnames(dat.psm)[3:12],
  cond = as.factor(cond)
)

gene.matrix = as.matrix(dat.gene.nm)
design = model.matrix(~cond,sampleTable)

fit1 <- eBayes(lmFit(gene.matrix,design))
# add PSM count for each gene
fit1$count <- psm.count.table[rownames(fit1$coefficients),2]

fit2 = spectraCounteBayes(fit1)
**VarianceBoxplot**

*generate a boxplot of the variance*

**Description**

This function is to draw a boxplot of the variance of genes quantified by different number of peptides/PSMs. Red curve indicate DEqMS prior variance.

**Usage**

```r
VarianceBoxplot(fit, n = 20, xlab = "count", ylab = "log(Variance)", main = "")
```

**Arguments**

- `fit` an object returned from `spectraCounteBayes` function
- `n` set a number to plot only the genes with count value smaller or equal to n
- `xlab` the title for x axis
- `ylab` the title for y axis
- `main` the title for the figure

**Value**

return a plot graphic

**Author(s)**

Yafeng Zhu

**Examples**

```r
library(ExperimentHub)
eh = ExperimentHub(localHub = TRUE)
query(eh, "DEqMS")
dat.psm = eh[["EH1663"]]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log, group_col = 2)

psm.count.table = as.data.frame(table(dat.psm$gene)) # generate PSM count table
rownames(psm.count.table) = psm.count.table$Var1

cond = c("ctrl", "miR191", "miR372", "miR519", "ctrl",
"miR372", "miR519", "ctrl", "miR191", "miR372")

sampleTable <- data.frame(
row.names = colnames(dat.psm)[3:12],
```
cond = as.factor(cond)

gene.matrix = as.matrix(dat.gene.nm)
design = model.matrix(~cond,sampleTable)

fit1 <- eBayes(lmFit(gene.matrix,design))
# add PSM count for each gene
fit1$count <- psm.count.table[rownames(fit1$coefficients),2]

fit2 = spectraCounteBayes(fit1)

VarianceBoxplot(fit2,xlab="PSM count",main="TMT data PXD004163")

---

**VarianceScatterplot**

*generate a scatter plot of the variance*

**Description**

This function is to draw a scatter plot of the variance against the number of quantified peptides/PSMs. Red curve indicate DEqMS prior variance.

**Usage**

```
VarianceScatterplot(fit, xlab="log2(count)",
ylab = "log(Variance)", main="")
```

**Arguments**

- `fit` an object returned from spectraCounteBayes function
- `xlab` the title for x axis
- `ylab` the title for y axis
- `main` the title for the figure

**Value**

return a plot graphic

**Author(s)**

Yafeng Zhu
Examples

library(ExperimentHub)
eh = ExperimentHub(localHub=TRUE)
query(eh, "DEqMS")
dat.psm = eh["EH1663"]

dat.psm.log = dat.psm
dat.psm.log[,3:12] = log2(dat.psm[,3:12])

dat.gene.nm = medianSweeping(dat.psm.log, group_col = 2)

psm.count.table = as.data.frame(table(dat.psm$gene)) # generate PSM count table
rownames(psm.count.table)=psm.count.table$Var1

cond = c("ctrl","miR191","miR372","miR519","ctrl",
"miR372","miR519","ctrl","miR191","miR372")

sampleTable <- data.frame(
row.names = colnames(dat.psm)[3:12],
cond = as.factor(cond)
)

gene.matrix = as.matrix(dat.gene.nm)
design = model.matrix(~cond,sampleTable)

fit1 <- eBayes(lmFit(gene.matrix,design))
# add PSM count for each gene
fit1$count <- psm.count.table[rownames(fit1$coefficients),2]

fit2 = spectraCounteBayes(fit1)

VarianceScatterplot(fit2,xlab="log2(PSM count)",main="TMT data PXD004163")
Index

equalMedianNormalization, 2
farmsSummary, 3
medianSummary, 4
medianSweeping, 5
medpolishSummary, 6
outputResult, 7
peptideProfilePlot, 8
Residualplot, 9
spectraCounteBayes, 10
VarianceBoxplot, 12
VarianceScatterplot, 13