Package: msqrob2 (via r-universe)

November 1, 2026

Title Robust statistical inference for quantitative LC-MS proteomics

Version 1.20.0

Description msqrob2 provides a robust linear mixed model framework for assessing differential abundance in MS-based Quantitative proteomics experiments. Our workflows can start from raw peptide intensities or summarised protein expression values. The model parameter estimates can be stabilized by ridge regression, empirical Bayes variance estimation and robust M-estimation. msqrob2's hurde workflow can handle missing data without having to rely on hard-to-verify imputation assumptions, and, outcompetes state-of-the-art methods with and without imputation for both high and low missingness. It builds on QFeature infrastructure for quantitative mass spectrometry data to store the model results together with the raw data and preprocessed data.

Depends R (>= 4.1), QFeatures (>= 1.1.2)

Imports stats, methods, lme4, purrr, BiocParallel, Matrix, MASS, limma, SummarizedExperiment, MultiAssayExperiment, codetools, matrixStats, ggplot2, assertthat, dplyr, grDevices, utils, rlang

Suggests stringr, ExploreModelMatrix, kableExtra, ComplexHeatmap, scater, multcomp, gridExtra, knitr, BiocStyle, RefManageR, sessioninfo, rmarkdown, testthat, tidyverse, tidyr, plotly, MsDataHub, MSnbase, MsCoreUtils, covr, arrow, data.table, ggcorrplot, iq

License Artistic-2.0

Collate 'msqrob-framework.R' 'allGenerics.R' 'calculateNormFactors.R' 'accessors.R' 'msqrob.R' 'msqrob-utils.R' 'StatModel-methods.R' 'hypothesisTest-methods.R' 'msqrob-methods.R' 'msqrobAggregate.R' 'topFeatures.R' 'data.R' 'msqrobQB.R' 'msqrobHurdle-methods.R' 'collect-plot-results.R'

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown=TRUE)

RoxygenNote 7.3.3

biocViews Proteomics, Metabolomics, MassSpectrometry, DifferentialExpression, MultipleComparison, Regression, ExperimentalDesign, Software, ImmunoOncology, Normalization, TimeCourse, Preprocessing

URL https://github.com/statOmics/msqrob2

BugReports https://github.com/statOmics/msqrob2/issues

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:56:09 UTC

RemoteUrl https://github.com/bioc/msqrob2

RemoteRef RELEASE_3_23

RemoteSha 3206088d43f7f5727728767a81cf923f3e5c00bd

Contents

.computeNfLogMedian . . . 3
.computeNfLogMedianOfRatios . . . 3
createPairwiseContrasts . . . 4
getContrast,StatModel-method . . . 5
getModel,StatModel-method . . . 6
hypothesisTest,SummarizedExperiment-method . . . 8
makeContrast . . . 11
msqrob,SummarizedExperiment-method . . . 12
msqrobAggregate,SummarizedExperiment-method . . . 15
msqrobCollect . . . 17
msqrobGlm . . . 19
msqrobHurdle,SummarizedExperiment-method . . . 20
msqrobLm . . . 22
msqrobLmer . . . 23
msqrobQB,SummarizedExperiment-method . . . 25
nfLogMedian,matrix-method . . . 27
nfLogMedianOfRatios,matrix-method . . . 29
pe . . . 32
plotVolcano . . . 33
smallestUniqueGroups . . . 34
StatModel-class . . . 35
topFeatures . . . 37

Index **39**

.computeNfLogMedian *Helper function to calculate sample-specific normalization factors on the log2 scale using conventional median normalisation*

Description

Helper function to calculate sample-specific normalization factors on the log2 scale using conventional median normalisation

Usage

```
.computeNfLogMedian(mat, na.rm = TRUE)
```

Arguments

mat	A matrix object.
na.rm	Logical; indicates if missing values are to be removed. Default is TRUE as missing values typically occur in proteomics data.

Details

Implementation assumes that the assay values are already on the log scale. The normalization factors are computed on the log scale.

Value

A numeric vector of log-scale normalization factors, one per sample (column).

.computeNfLogMedianOfRatios *Helper function to calculate sample-specific normalization factors*

Description

Helper function to calculate sample-specific normalization factors on the log2 scale using a median-of-ratios approach similar to that used in DESeq2 for bulk RNA-seq data.

The method proceeds as follows:

1. A pseudo-reference sample is constructed as the row-wise mean of the log2 intensities (equivalent to the log2-transformed geometric mean).
2. For each sample, log2 ratios relative to pseudo-reference are computed.
3. The normalization factor for each sample is obtained as the median of these log2 ratios (column-wise median).

Usage

```
.computeNfLogMedianOfRatios(mat, na.rm = TRUE)
```

Arguments

mat	A matrix object.
na.rm	Logical; indicates if missing values are to be removed. Default is TRUE as missing values typically occur in proteomics data.

Details

This implementation assumes that assay values are already on the log scale. The normalization factors are computed on the log scale.

Value

A numeric vector of log2-scale normalization factors, one per sample (column).

---

createPairwiseContrasts

Construct all contrasts for all pairwise comparisons between all levels of a factor

---

Description

Construct all contrasts for all pairwise comparisons between all levels of a factor.

Usage

```
createPairwiseContrasts(
  formula,
  coldata,
  var,
  ridge = FALSE,
  nullHypothesis = " = 0"
)
```

Arguments

formula	Model formula. The model is built based on the covariates in the 'coldata' object.
coldata	data.frame or DFrame with information on the design.
var	object of type 'character' with the name of the factor in the formula for which the pairwise comparisons will be made.
ridge	logical indicating if the msqrob models are fitted with or without ridge regression. The default is 'ridge = FALSE'
nullHypothesis	object of type character that specifies the value of the contrast under the null hypothesis. The default is ' = 0'.

Value

Vector of type character with nulhypotheses for all contrasts.

Examples

```
# Load example data
# The data are a Feature object containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Model with intercept (conditiona is reference class)
createPairwiseContrasts(~ condition, colData(pe), "condition")

# Model without intercept
createPairwiseContrasts(~ -1 + condition, colData(pe), "condition")
```

---

getContrast,StatModel-method

Methods for StatModel class

Description

Methods for StatModel class

getContrast(object, L) to calculate contrasts of the model parameters

varContrast(object, L) to calculate the variance-covariance matrix of the contrasts

Usage

```
## S4 method for signature 'StatModel'
getContrast(object, L)

## S4 method for signature 'StatModel'
varContrast(object, L)
```

Arguments

object	A list with elements of the class StatModel that are estimated using the msqrob function
L	contrast numeric matrix specifying one or more contrasts of the linear model coefficients to be tested equal to zero. The rownames of the matrix should be equal to the names of parameters of the model.

Value

A matrix with the calculated contrasts or variance-covariance matrix of contrasts

Examples

```
data(pe)
# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)

# Define contrast
getCoef(rowData(pe[["protein"]])$msqrobModels[[1]])
# Define contrast for log2 fold change between condition c and condition b:
L <- makeContrast("conditionc - conditionb=0", c("conditionb", "conditionc"))

getContrast(rowData(pe[["protein"]])$msqrobModels[[1]], L)
varContrast(rowData(pe[["protein"]])$msqrobModels[[1]], L)
```

---

getModel,StatModel-method

Accessor functions for StatModel class

Description

Accessor functions for StatModel class

getModel(object) to get model

getFitMethod(object) to get the parameter estimation method

getCoef(object) to get the parameter estimates of the mean model

getDF(object) to get the residual degrees of freedom of the model

getVar(object) to get the residual variance of the model

getSigma(object) to get the residual standard deviation of the model

getDfPosterior(object) to get the degrees of freedom of the empirical Bayes variance estimator

getVarPosterior(object) to get the empirical Bayes variance

getSigmaPosterior(object) to get the empirical Bayes standard deviation

getVcovUnscaled(object) to get the unscaled variance covariance matrix of the model parameters

Usage

```
## S4 method for signature 'StatModel'
getModel(object)

## S4 method for signature 'StatModel'
getFitMethod(object)

## S4 method for signature 'StatModel'
getCoef(object)

## S4 method for signature 'StatModel'
getDfPosterior(object)

## S4 method for signature 'StatModel'
getVarPosterior(object)

## S4 method for signature 'StatModel'
getSigmaPosterior(object)

## S4 method for signature 'StatModel'
getDF(object)

## S4 method for signature 'StatModel'
getVar(object)

## S4 method for signature 'StatModel'
getSigma(object)

## S4 method for signature 'StatModel'
getVcovUnscaled(object)
```

Arguments

object	StatModel object

Value

The requested parameter of the StatModel object

Examples

```
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)
getCoef(rowData(pe[["protein"]])$msqrobModels[[1]])
```

```
getModel(rowData(pe[["protein"]])$msqrobModels[[1]])
getFitMethod(rowData(pe[["protein"]])$msqrobModels[[1]])
# Similar for the remaining accessors
```

---

hypothesisTest,SummarizedExperiment-method

Parameter estimates, standard errors and statistical inference on differential expression analysis

---

Description

Summary table of the estimates for differential expression of features

Usage

```
## S4 method for signature 'SummarizedExperiment'
hypothesisTest(
  object,
  contrast,
  adjust.method = "BH",
  modelColumn = "msqrobModels",
  resultsColumnNamePrefix = "",
  overwrite = FALSE
)

## S4 method for signature 'SummarizedExperiment'
hypothesisTestHurdle(
  object,
  contrast,
  adjust.method = "BH",
  modelColumn = "msqrobHurdle",
  resultsColumnNamePrefix = "hurdle_",
  overwrite = FALSE
)

## S4 method for signature 'QFeatures'
hypothesisTest(
  object,
  i,
  contrast,
  adjust.method = "BH",
  modelColumn = "msqrobModels",
  resultsColumnNamePrefix = "",
  overwrite = FALSE
)

## S4 method for signature 'QFeatures'
```

```
hypothesisTestHurdle(
  object,
  i,
  contrast,
  adjust.method = "BH",
  modelColumn = "msqrobHurdle",
  resultsColumnNamePrefix = "hurdle_",
  overwrite = FALSE
)
```

Arguments

object	SummarizedExperiment or QFeatures instance
contrast	numeric matrix specifying one or more contrasts of the linear model coefficients to be tested equal to zero. If multiple contrasts are given (multiple columns) then results will be returned for each contrast. The rownames of the matrix should be equal to the names of parameters of the model that are involved in the contrast. The column names of the matrix will be used to construct names to store the results in the rowData of the SummarizedExperiment or of the assay of the QFeatures object. The contrast matrix can be made using the makeContrast function.
adjust.method	character specifying the method to adjust the p-values for multiple testing. Options, in increasing conservatism, include '"none"', '"BH"', '"BY"' and '"holm"'. See 'p.adjust' for the complete list of options. Default is "BH" the Benjamini-Hochberg method to controle the False Discovery Rate (FDR).
modelColumn	character to indicate the variable name that was used to store the msqrob models in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "msqrobModels" when the hypothesisTest function is used and "msqrobHurdle" for hypothesisTestHurdle.
resultsColumnNamePrefix	character to indicate the the prefix for the variable name that will be used to store test results in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "" so that the variable name with the results will be the column name of the column in the contrast matrix L. If L is a matrix with multiple columns, multiple results columns will be made, one for each contrast. If L is a matrix with a single column which has no column names and if resultsColumnNamePrefix="" the results will be stored in the column with name msqrobResults. For hypothesisTestHurdle the default prefix is "hurdle_". If L is a matrix with one column and has no column names and if resultsColumnNamePrefix="hurdle_" the results will be stored in the column with name hurdleResults.
overwrite	boolean(1) to indicate if the column in the rowData has to be overwritten if the modelColumnName already exists. Default is FALSE.
i	character or integer to specify the element of the QFeatures that contains the log expression intensities that will be modelled.

Value

A SummarizedExperiment or a `QFeatures` instance augmented with the test results.

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)

# Define contrast
getCoef(rowData(pe[["protein"]])$msqrobModels[[1]])
# Assess log2 fold change between condition c and condition b
L <- makeContrast(
    "conditionc - conditionb=0",
    c("conditionb", "conditionc")
)

# example SummarizedExperiment instance
se <- pe[["protein"]]
se <- hypothesisTest(se, L)
head(rowData(se)$"conditionc - conditionb", 10)
# Volcano plot
plot(-log10(pval) ~ logFC,
    rowData(se)$"conditionc - conditionb",
    col = (adjPval < 0.05) + 1
)

# Example for QFeatures instance
# Assess log2 fold change between condition b and condition a (reference class),
# condition c and condition a, and, condition c and condition b.
L <- makeContrast(
    c(
        "conditionb=0",
        "conditionc=0",
        "conditionc - conditionb=0"
    ),
    c("conditionb", "conditionc")
)
```

```
pe <- hypothesisTest(pe, i = "protein", L)
head(rowData(pe[["protein"]])$"conditionb", 10)
# Volcano plots
par(mfrow = c(1, 3))
plot(-log10(pval) ~ logFC,
    rowData(pe[["protein"]])$"conditionb",
    col = (adjPval < 0.05) + 1,
    main = "log2 FC b-a"
)
plot(-log10(pval) ~ logFC,
    rowData(pe[["protein"]])$"conditionc",
    col = (adjPval < 0.05) + 1,
    main = "log2 FC c-a"
)
plot(-log10(pval) ~ logFC,
    rowData(pe[["protein"]])$"conditionc - conditionb",
    col = (adjPval < 0.05) + 1,
    main = "log2 FC c-b"
)

# Hurdle method
pe <- msqrobHurdle(pe, i = "protein", formula = ~condition)
pe <- hypothesisTestHurdle(pe, i = "protein", L)
head(rowData(pe[["protein"]])$"hurdle_conditionb", 10)
```

`makeContrast` *Make contrast matrix*

Description

Construct the contrast matrix corresponding to specified contrasts of a set of parameters.

Usage

```
makeContrast(contrasts, parameterNames)
```

Arguments

`contrasts`	character vector specifying contrasts, i.e. the linear combination of the model-parameters that equals to zero.
`parameterNames`	character vector specifying the model parameters that are involved in the contrasts, e.g if we model data of three conditions using a factor condition with three levels a, b and c then our model will have 3 mean parameters named (Intercept), conditionb and conditionc. Hence the log2 fold change between b and a is conditionb. Under the null hypothesis the log2 fold change equals 0. Which is to be encoded as "conditionb=0". If we would like to test for log2 fold change between condition c and b we assess if the log2 fold change conditionc-conditionb equals 0, encoded as "condtionb-conditionc=0".

Value

A numeric contrast matrix with rownames that equal the model parameters that are involved in the contrasts

Examples

```
makeContrast(c("conditionb = 0"),
    parameterNames = c(
        "(Intercept)",
        "conditionb",
        "conditionc"
    )
)
makeContrast(c("conditionc=0"),
    parameterNames = c("conditionc")
)
makeContrast(c(
    "conditionb=0",
    "conditionc=0",
    "conditionc-conditionb=0"
),
parameterNames = c(
    "conditionb",
    "conditionc"
)
)
```

---

msqrob,SummarizedExperiment-method

Methods to fit msqrob models with ridge regression and/or random effects using lme4

Description

Parameter estimation of msqrob models for `QFeatures` and `SummarizedExperiment` instance.

Usage

```
## S4 method for signature 'SummarizedExperiment'
msqrob(
  object,
  formula,
  modelColumnName = "msqrobModels",
  overwrite = FALSE,
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  tol = 1e-06,
```

```
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE))
)

## S4 method for signature 'QFeatures'
msqrob(
  object,
  i,
  formula,
  modelColumnName = "msqrobModels",
  overwrite = FALSE,
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  tol = 1e-06,
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE))
)
```

Arguments

object	SummarizedExperiment or QFeatures instance
formula	Model formula. The model is built based on the covariates in the data object.
modelColumnName	character to indicate the variable name that is used to store the msqrob models in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "msqrobModels".
overwrite	boolean(1) to indicate if the column in the rowData has to be overwritten if the modelColumnName already exists. Default is FALSE.
robust	boolean(1) to indicate if robust regression is performed to account for outliers. Default is TRUE. If FALSE an OLS fit is performed.
ridge	boolean(1) to indicate if ridge regression is performed. Default is FALSE. If TRUE the fixed effects are estimated via penalized regression and shrunken to zero.
maxitRob	numeric(1) indicating the maximum iterations in the IRWLS algorithm used in the M-estimation step of the robust regression.
tol	numeric(1) indicating the tolerance for declaring convergence of the M-estimation loop.
doQR	boolean(1) to indicate if QR decomposition is used when adopting ridge regression. Default is TRUE. If FALSE the predictors of the fixed effects are not transformed, and the degree of shrinkage can depend on the encoding.
lmerArgs	a list (of correct class, resulting from 'lmerControl()' containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer, see the 'lmerControl' documentation of the lme4 package for more details. Default is list(control = lmerControl(calc.derivs = FALSE))

i — character or integer to specify the element of the QFeatures that contains the log expression intensities that will be modelled.

Value

A SummarizedExperiment or a QFeatures instance with the models.

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit MSqrob model using robust linear regression upon summarization of
# peptide intensities into protein expression values.
# For summarized SummarizedExperiment
se <- pe[["protein"]]
se
colData(se) <- colData(pe)
se <- msqrob(se, formula = ~condition, modelColumnName = "rlm")
getCoef(rowData(se)$rlm[[1]])

# For features object
pe <- msqrob(pe, i = "protein", formula = ~condition, modelColumnName = "rlm")
# with ridge regression (slower)
pe <- msqrob(pe, i = "protein", formula = ~condition, ridge = TRUE, modelColumnName = "ridge")

# compare for human protein (no DE)==> large shrinkage to zero
cbind(getCoef(rowData(pe[["protein"]])$rlm[[1]]), getCoef(rowData(pe[["protein"]])$ridge[[1]]))

# compare for ecoli protein (DE)==> almost no shrinkage to zero
cbind(
    getCoef(rowData(pe[["protein"]])$rlm[["P00956"]]),
    getCoef(rowData(pe[["protein"]])$ridge[["P00956"]])
)
```

msqrobAggregate,SummarizedExperiment-method

Method to fit msqrob models with robust regression and/or ridge regression and/or random effects It models multiple features simultaneously, e.g. multiple peptides from the same protein.

Description

Parameter estimation of msqrob models for `QFeatures`instance. The method aggregates features within the model e.g. from peptides to proteins. It provides fold change estimates and their associated uncertainty at the aggregated level (e.g. protein level) while correcting for the peptide species that are observed in each sample. It also addresses the correlation in the data, e.g. the peptide data for the same protein in a sample are correlate because they originate from the same protein pool. The method however does not return aggregated expression values for each sample. For visualisation purposes aggregated expression values are provide by the `aggregateFeatures` function from the `QFeatures` Package

Usage

```
## S4 method for signature 'SummarizedExperiment'
msqrobAggregate(
  object,
  formula,
  fcol,
  aggregateFun = MsCoreUtils::robustSummary,
  modelColumnName = "msqrobModels",
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  tol = 1e-06,
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE))
)

## S4 method for signature 'QFeatures'
msqrobAggregate(
  object,
  formula,
  i,
  fcol,
  name = "msqrobAggregate",
  aggregateFun = MsCoreUtils::robustSummary,
  modelColumnName = "msqrobModels",
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  tol = 1e-06,
```

```
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE))
)
```

Arguments

object	QFeatures instance
formula	Model formula. The model is built based on the covariates in the data object.
fcol	The feature variable of assay 'i' defining how to summarise the features.
aggregateFun	A function used for quantitative feature aggregation. Details can be found in the documentation of the aggregateFeatures of the QFeatures package.
modelColumnName	character to indicate the variable name that is used to store the msqrob models in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "msqrobModels".
robust	boolean(1) to indicate if robust regression is performed to account for outliers. Default is TRUE. If FALSE an OLS fit is performed.
ridge	boolean(1) to indicate if ridge regression is performed. Default is FALSE. If TRUE the fixed effects are estimated via penalized regression and shrunken to zero.
maxitRob	numeric(1) indicating the maximum iterations in the IRWLS algorithm used in the M-estimation step of the robust regression.
tol	numeric(1) indicating the tolerance for declaring convergence of the M-estimation loop.
doQR	boolean(1) to indicate if QR decomposition is used when adopting ridge regression. Default is TRUE. If FALSE the predictors of the fixed effects are not transformed, and the degree of shrinkage can depend on the encoding.
lmerArgs	a list (of correct class, resulting from 'lmerControl()' containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer, see the 'lmerControl' documentation of the lme4 package for more details. Default is list(control = lmerControl(calc.derivs = FALSE))
i	character or integer to specify the element of the QFeatures that contains the log expression intensities that will be modelled.
name	A 'character(1)' naming the new assay. Default is 'newAssay'. Note that the function will fail if there's already an assay with 'name'.

Value

A 'QFeatures' object with an additional assay.

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Fit MSqrob model using robust ridge regression starting from peptide intensities
# The fold changes are calculated at the protein level while correcting for
# the different peptide species in each sample and the correlation between
# peptide intensities of peptides of the same protein in the same sample.
colData(pe)$samples <- rownames(colData(pe))
pe <- msqrobAggregate(pe, i = "peptide", fcol = "Proteins",
     formula = ~condition + (1|samples) + (1|Sequence),
     ridge = TRUE)
getCoef(rowData(pe[["msqrobAggregate"]])$msqrobModels[["P00956"]])

## Same but on a SummarizedExperiment object
se <- getWithColData(pe, "peptide")
se <- msqrobAggregate(se, fcol = "Proteins",
                      formula = ~condition + (1|samples) + (1|Sequence),
                      ridge = TRUE)
getCoef(rowData(se)$msqrobModels[["P00956"]])
```

`msqrobCollect` *Function to collect the inference tables generated by the msqrob2 statistical inference workflow.*

Description

Function to collect the inference tables generated by the msqrob2 statistical inference workflow.

Usage

```
msqrobCollect(object, contrast, resultsColumnNamePrefix = "", combine = TRUE)
```

Arguments

`object`	A `SummarizedExperiment` object containing model inference performed by `hypothesisTest()`.
`contrast`	A numeric `matrix` providing the contrast matrix used during model estimation or a character `vector` providing the names of the contrast used during model estimation.
`resultsColumnNamePrefix`	A `character(1)` providing the prefix used when running `hypothesisTest()`.

combine A logical(1) indicating whether the result tables should be combined in a single table (default) or return as a list of tables. When combined, two new variables are created: 1. contrasttells from which contrast (column of the contrast matrix) the results where obtained; 2. feature provides the name of the modelled feature (this information is taken from the rownames of the tables, but this are made unique upon combining).

Value

Result tables for contrasts combined in a single table or list

Examples

```
## Load example data
## The data are a Feature object containing
## a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
## The data are a subset of spike-in the human-ecoli study
## The variable condition in the colData of the Feature object
## contains information on the spike in condition a-e (from low to high)
data(pe)

## Aggregate peptide intensities in protein expression values
## Note that the peptide intensities were already normalised!
pe <- aggregateFeatures(pe,
                        i = "peptide",
                        fcol = "Proteins",
                        name = "protein")

## Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)

## Define contrast
design <- model.matrix(~condition, data = colData(pe))

## Assess log2 fold change between reference condition a and condition b
L <- makeContrast(
    contrasts = c("conditionb = 0",
                  "conditionc = 0",
                  "conditionc - conditionb = 0"),
    parameterNames = colnames(design))


pe <- hypothesisTest(pe, i = "protein", L)

## Extract Results
inference <- msqrobCollect(pe[["protein"]], L)
head(inference)
```

`msqrobGlm` *Function to fit msqrob models to peptide counts using glm*

Description

Low-level function for parameter estimation with msqrob by modeling peptide counts using quasi-binomial glm

Usage

```
msqrobGlm(y, npep, formula, data, priorCount = 0.1, binomialBound = TRUE)
```

Arguments

`y`	A `matrix` with the peptide counts. The features are along the rows and samples along the columns.
`npep`	A vector with number of peptides per protein. It has as length the number of rows of y. The counts are equal or larger than the largest peptide count in y.
`formula`	Model formula. The model is built based on the covariates in the data object.
`data`	A `DataFrame` with information on the design. It has the same number of rows as the number of columns (samples) of `y`.
`priorCount`	A 'numeric(1)', which is a prior count to be added to the observations to shrink the estimated log-fold-changes towards zero.
`binomialBound`	logical, if 'TRUE' then the quasibinomial variance estimator will be never smaller than 1 (no underdispersion).

Value

A list of objects of the `StatModel` class.

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")
pe
```

```
# Fit MSqrob model using robust regression with the MASS rlm function
models <- msqrobGlm(
    aggcounts(pe[["protein"]]),
    rowData(pe[["protein"]])[[".n"]],
    ~condition,
    colData(pe)
)
getCoef(models[[1]])
```

`msqrobHurdle,SummarizedExperiment-method`

Function to fit msqrob hurdle models

Description

Fitting a hurdle msqrob model with an intensity component for assessing differential abundance and an count component that is modeling peptide counts using quasibinomial glm for differential detection of the number of features that were not missing and used for aggregation

Usage

```
## S4 method for signature 'SummarizedExperiment'
msqrobHurdle(
  object,
  formula,
  modelColumnName = "msqrobHurdle",
  overwrite = FALSE,
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  tol = 1e-06,
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE)),
  priorCount = 0.1,
  binomialBound = TRUE
)

## S4 method for signature 'QFeatures'
msqrobHurdle(
  object,
  i,
  formula,
  modelColumnName = "msqrobHurdle",
  overwrite = FALSE,
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
```

```
  tol = 1e-06,
  doQR = TRUE,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE)),
  priorCount = 0.1,
  binomialBound = TRUE
)
```

Arguments

object	SummarizedExperiment or QFeatures instance with an assay that is generated with the aggreateFeatures function from the QFeatures package
formula	Model formula. Both model components are built based on the covariates in the data object.
modelColumnName	character to indicate the variable name that is used to store the msqrob models in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "msqrobHurdle".
overwrite	boolean(1) to indicate if the column in the rowData has to be overwritten if the modelColumnName already exists. Default is FALSE.
robust	boolean(1) to indicate if robust regression is performed to account for outliers when fitting the intensity component of the hurdle model. Default is TRUE. If FALSE an OLS fit is performed.
ridge	boolean(1) to indicate if ridge regression is performed. Default is FALSE. If TRUE the fixed effects of the intensity component of the hurdle model are estimated via penalized regression and shrunken to zero.
maxitRob	numeric(1) indicating the maximum iterations in the IRWLS algorithm used in the M-estimation step of the robust regression for fitting the intensity component of the hurdle model.
tol	numeric(1) indicating the tolerance for declaring convergence of the M-estimation loop of the intensity component of the hurdle model.
doQR	boolean(1) to indicate if QR decomposition is used when adopting ridge regression for the intensity component of the model. Default is TRUE. If FALSE the predictors of the fixed effects are not transformed, and the degree of shrinkage can depend on the encoding.
lmerArgs	a list (of correct class, resulting from 'lmerControl()' containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer, see the 'lmerControl' documentation of the lme4 package for more details. Default is list(control = lmerControl(calc.derivs = FALSE))
priorCount	A 'numeric(1)', which is a prior count to be added to the observations to shrink the estimated odds ratios of the count component towards zero. Default is 0.1.
binomialBound	logical, if 'TRUE' then the quasibinomial variance estimator will be never smaller than 1 (no underdispersion). Default is TRUE.
i	character or integer to specify the element of the QFeatures that contains the log expression intensities that will be modelled.

Value

SummarizedExperiment or QFeatures instance

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities to protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit Hurdle MSqrob model
# For summarized SummarizedExperiment
se <- pe["protein"]
se
colData(se) <- colData(pe)
se <- msqrobHurdle(se, formula = ~condition)
getCoef(rowData(se)$msqrobHurdleIntensity[[1]])
getCoef(rowData(se)$msqrobHurdleCount[[1]])

# For features object
pe <- msqrobHurdle(pe, i = "protein", formula = ~condition)
getCoef(rowData(pe["protein"])$msqrobHurdleIntensity[[1]])
getCoef(rowData(pe["protein"])$msqrobHurdleCount[[1]])
```

`msqrobLm` *Function to fit msqrob models using lm and rlm*

Description

Low-level function for parameter estimation with msqrob using the ordinary least squares or robust regression base on the MASS::rlm function.

Usage

```
msqrobLm(y, formula, data, robust = TRUE, maxitRob = 5)
```

Arguments

y	A matrix with the quantified feature intensities. The features are along the rows and samples along the columns.
formula	Model formula. The model is built based on the covariates in the data object.
data	A DataFrame with information on the design. It has the same number of rows as the number of columns (samples) of y.
robust	boolean(1) to indicate if robust regression is performed to account for outliers. Default is TRUE. If FALSE an OLS fit is performed.
maxitRob	numeric(1) indicating the maximum iterations in the IRWLS algorithm used in the M-estimation step of the robust regression.

Value

A list of objects of the StatModel class.

Author(s)

Lieven Clement, Oliver M. Crook

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")
pe

# Fit MSqrob model using robust regression with the MASS rlm function
models <- msqrobLm(assay(pe[["protein"]]), ~condition, colData(pe))
#' getCoef(models[[1]])
```

---

msqrobLmer *Function to fit msqrob models with ridge regression and/or random effects using lme4*

Description

Low-level function for parameter estimation with msqrob using the robust ridge regression. The models can be fitted for each feature (e.g. summarised protein expression values) or multiple features belonging to the same accession can be modelled simultaneously e.g. peptide-based models where all peptide intensities for the same protein are modelled simultaneously. The fold changes and uncertainty estimates are then calculated at the protein level while correcting for peptide species and within sample correlation.

Usage

```
msqrobLmer(
  y,
  formula,
  data,
  rowdata = NULL,
  tol = 1e-06,
  robust = TRUE,
  ridge = FALSE,
  maxitRob = 1,
  doQR = TRUE,
  featureGroups = NULL,
  lmerArgs = list(control = lmerControl(calc.derivs = FALSE))
)
```

Arguments

y	A matrix with the quantified feature intensities. The features are along the rows and samples along the columns.
formula	Model formula. The model is built based on the covariates in the data object.
data	A DataFrame with information on the design. It has the same number of rows as the number of columns (samples) of y.
rowdata	A DataFrame with the rowData information of the SummarizedExperiment. It has the same number of rows as the number of rows (features) of y.
tol	numeric(1) indicating the tolerance for declaring convergence of the M-estimation loop.
robust	boolean(1) to indicate if robust regression is performed to account for outliers. Default is TRUE. If FALSE an OLS fit is performed.
ridge	boolean(1) to indicate if ridge regression is performed. Default is FALSE. If TRUE the fixed effects are estimated via penalized regression and shrunken to zero.
maxitRob	numeric(1) indicating the maximum iterations in the IRWLS algorithm used in the M-estimation step of the robust regression.
doQR	boolean(1) to indicate if QR decomposition is used when adopting ridge regression. Default is TRUE. If FALSE the predictors of the fixed effects are not transformed, and the degree of shrinkage can depend on the encoding.
featureGroups	vector of type character or vector of type factor indicating how to aggregate the features. Is only used when multiple features are used to build the model, e.g. when starting from peptide data and modelling the fold change at the protein level. The default is NULL
lmerArgs	a list (of correct class, resulting from 'lmerControl()' containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer, see the 'lmerControl' documentation of the lme4 package for more details. Default is list(control = lmerControl(calc.derivs = FALSE))

Value

A list of objects of the `StatModel` class.

Author(s)

Lieven Clement, Oliver M. Crook

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit MSqrob model using robust ridge regression upon summarization of
# peptide intensities into protein expression values
modelsRidge <- msqrobLmer(assay(pe[["protein"]]), ~condition, data = colData(pe),
                          ridge = TRUE)
getCoef(modelsRidge[[1]])

# Fit MSqrob model using robust ridge regression starting from peptide intensities
# The fold changes are calculated at the protein level while correcting for
# the different peptide species in each sample and the correlation between
# peptide intensities of peptides of the same protein in the same sample.
# Add the samples variable to colData
colData(pe)$samples <- rownames(colData(pe))
modelsPepBased <- msqrobLmer(assay(pe[["peptide"]]),
    formula = ~condition + (1|samples) + (1|Sequence), data = colData(pe),
   rowdata = rowData(pe[["peptide"]]), featureGroups = rowData(pe[["peptide"]])$Proteins,
    ridge = TRUE)
getCoef(modelsPepBased[[1]])
```

`msqrobQB,SummarizedExperiment-method`

Function to fit msqrob models to peptide counts using glm

Description

Low-level function for parameter estimation with msqrob by modeling peptide counts using quasi-binomial glm

Usage

```
## S4 method for signature 'SummarizedExperiment'
msqrobQB(
  object,
  formula,
  modelColumnName = "msqrobQbModels",
  overwrite = FALSE,
  priorCount = 0.1,
  binomialBound = TRUE
)

## S4 method for signature 'QFeatures'
msqrobQB(
  object,
  i,
  formula,
  modelColumnName = "msqrobQbModels",
  overwrite = FALSE,
  priorCount = 0.1,
  binomialBound = TRUE
)
```

Arguments

`object`	`SummarizedExperiment` or `QFeatures` instance
`formula`	Model formula. The model is built based on the covariates in the data object.
`modelColumnName`	`character` to indicate the variable name that is used to store the msqrob models in the rowData of the SummarizedExperiment instance or of the assay of the QFeatures instance. Default is "msqrobModels".
`overwrite`	`boolean(1)` to indicate if the column in the rowData has to be overwritten if the modelColumnName already exists. Default is FALSE.
`priorCount`	A 'numeric(1)', which is a prior count to be added to the observations to shrink the estimated log-fold-changes towards zero. Default is 0.1.
`binomialBound`	logical, if 'TRUE' then the quasibinomial variance estimator will be never smaller than 1 (no underdispersion). Default is TRUE.
`i`	`character` or `integer` to specify the element of the `QFeatures` that contains the log expression intensities that will be modelled.

Value

SummarizedExperiment or QFeatures instance

Author(s)

Lieven Clement

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate by counting how many peptide we observe for each protein
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit MSqrob model to peptide counts using a quasi-binomial model
# For summarized SummarizedExperiment
se <- pe[["protein"]]
se
colData(se) <- colData(pe)
se <- msqrobQB(se, formula = ~condition)
getCoef(rowData(se)$msqrobQbModels[[1]])

# For features object
pe <- msqrobQB(pe, i = "protein", formula = ~condition)
```

---

nfLogMedian,matrix-method

Methods to computes sample-specific normalization factors on the log scale using conventional median summarisation.

Description

Methods to computes sample-specific normalization factors on the log scale using conventional median summarisation.

Usage

```
## S4 method for signature 'matrix'
nfLogMedian(object, na.rm = TRUE)

## S4 method for signature 'SummarizedExperiment'
nfLogMedian(object, i, na.rm = TRUE)

## S4 method for signature 'QFeatures'
nfLogMedian(object, i, na.rm = TRUE)
```

Arguments

object	matrix, SummarizedExperiment or QFeatures object.

na.rm	Logical; indicates if missing values are to be removed. Default is TRUE as missing values typically occur in proteomics data.
i	An integer or character specifying which assay to use, only needed when object is SummarizedExperiment or QFeatures

Details

Implementation assumes that the assay values are already on the log scale. The normalization factors are computed on the log scale.

Value

A numeric vector of log2-scale normalization factors, one per sample (column).

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

##############################
### Example on QFeatures object
##############################

data(pe)

# Calculate log2 norm factor
nf_log <- nfLogMedian(pe, i="peptide")
nf_log

# Normalise peptide level data
pe <- sweep(pe,
    MARGIN = 2,
    STATS = nf_log,
    i = "peptide",
    name = "peptide_norm")

# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(assay(pe,"peptide"))
boxplot(assay(pe,"peptide_norm"))

##############################
### Example on SummarizedExperiment object
##############################

data(pe)
```

```
# Extract a summarised experiment from QFeatures object pe
se <- getWithColData(pe, i="peptide")

# Calculate log2 norm factor
nf_log <- nfLogMedian(se, i=1)
nf_log

# Normalise peptide level data and store it as a new assay in
# the SummarizedExperiment object
assays(se)[["peptide_norm"]] <- sweep(assay(se),
    MARGIN = 2,
    STATS = nf_log)

# Also give first assay a name
assayNames(se)[1] <- "peptide"

# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(assay(se,"peptide"))
boxplot(assay(se,"peptide_norm"))

################################
### Example on matrix object
################################

data(pe)

# Extract log2 transformed intensity  matrix from QFeatures object pe
mat <- assay(pe,"peptide")

# Calculate Norm factors
nf <- nfLogMedian(mat)
nf

# Normalise peptide level data
matnorm <- sweep(mat,
    MARGIN = 2,
    STATS = nf_log)
# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(mat)
boxplot(matnorm)
```

---

nfLogMedianOfRatios,matrix-method

Methods to calculate log-scale normalization factors using the median-of-ratios method

---

Description

Computes sample-specific normalization factors on the log2 scale using a median-of-ratios approach similar to that used in DESeq2 for bulk RNA-seq data.

The method proceeds as follows:

1. A pseudo-reference sample is constructed as the row-wise mean of the log2 intensities (equivalent to the log2-transformed geometric mean).
2. For each sample, log2 ratios relative to pseudo-reference are computed.
3. The normalization factor for each sample is obtained as the median of these log2 ratios (column-wise median).

Usage

```
## S4 method for signature 'matrix'
nfLogMedianOfRatios(object, na.rm = TRUE)

## S4 method for signature 'SummarizedExperiment'
nfLogMedianOfRatios(object, i, na.rm = TRUE)

## S4 method for signature 'QFeatures'
nfLogMedianOfRatios(object, i, na.rm = TRUE)
```

Arguments

`object`	`matrix`, `SummarizedExperiment` or `QFeatures` object.
`na.rm`	Logical; indicates if missing values are to be removed. Default is `TRUE` as missing values typically occur in proteomics data.
`i`	An integer or character specifying which assay to use, only needed when object is `SummarizedExperiment` or `QFeatures`

Details

This implementation assumes that the assay values are already on the log scale. The normalization factors are computed on the log scale.

Value

A numeric vector of log2-scale normalization factors, one per sample (column).

References

Love, M.I., Huber, W., Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15(12), 550.

Examples

```
# Load example data
# The data are a Feature object with containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

################################
### Example on QFeatures object
################################

data(pe)

# Calculate log2 norm factor
nf_log <- nfLogMedianOfRatios(pe, i="peptide")
nf_log

# Normalise peptide level data
pe <- sweep(pe,
    MARGIN = 2,
    STATS = nf_log,
    i = "peptide",
    name = "peptide_norm")

# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(assay(pe,"peptide"))
boxplot(assay(pe,"peptide_norm"))

################################
### Example on SummarizedExperiment object
################################

data(pe)

# Extract a summarised experiment from QFeatures object pe
se <- getWithColData(pe, i="peptide")

# Calculate log2 norm factor
nf_log <- nfLogMedianOfRatios(se, i=1)
nf_log

# Normalise peptide level data and store it as a new assay in
# the SummarizedExperiment object
assays(se)[["peptide_norm"]] <- sweep(assay(se),
    MARGIN = 2,
    STATS = nf_log)

# Also give first assay a name
assayNames(se)[1] <- "peptide"
```

```
# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(assay(se,"peptide"))
boxplot(assay(se,"peptide_norm"))

##############################
### Example on matrix object
##############################

data(pe)

mat <- assay(pe,"peptide")

nf <- nfLogMedianOfRatios(mat)
nf

# Normalise peptide level data
matnorm <- sweep(mat,
    MARGIN = 2,
    STATS = nf_log)
# Evaluate normalisation
par(mfrow=c(1,2))
boxplot(mat)
boxplot(matnorm)
```

---

pe *Example data for 100 proteins*

Description

Subset of peptides from 100 proteins from a quantitative mass spectrometry based proteomics dataset (PRIDE identifier: PXD003881 Shen et al. (2018)). E. Coli lysates were spiked at five different concentrations (3%, 4.5%, 6%, 7.5% and 9%wt/wt) in a stable human background (4 repl. per treatment). The twenty resulting samples were run on an Orbitrap Fusion mass spectrometer. Raw data files were processed with MaxQuant (version 1.6.1.0, Cox and Mann (2008)) using default search settings unless otherwise noted. Spectra were searched against the UniProtKB/SwissProt human and E. Coli reference proteome databases (07/06/2018), concatenated with the default Maxquant contaminant database. Carbamidomethylation of Cystein was set as a fixed modification, and oxidation of Methionine and acetylation of the protein amino-terminus were allowed as variable modifications. In silico cleavage was set to use trypsin/P, allowing two miscleavages. Match between runs was also enabled using default settings. The resulting peptide-to-spectrum matches (PSMs) were filtered by MaxQuant at 1% FDR.

Usage

```
data(pe)
```

Format

Feature set with an instance "peptide":

assay contains the raw peptide intensities

rowData contains a variable "Proteins" with the protein accession and an variable ecoli to indicate if the protein is a spikin.

colData contains a factor condition indicating the spike-in condition

Examples

```
data(pe)
head(colData(pe))
head(rowData(pe))
head(assay(pe))
```

---

plotVolcano *Volcano plot*

Description

`plot_volcano` generates a volcano plot for the results of hypothesis tests generated with the msqrob2 hypothesisTest method.

Usage

```
plotVolcano(
  resultsTable,
  significanceLevel = 0.05,
  significance_colors = c(`FALSE` = "black", `TRUE` = "red"),
  opacity = 0.5
)
```

Arguments

`resultsTable`	data frame returned by the msqrob2 hypothesisTest method
`significanceLevel`	nominal FDR-level at which features are considered significant (value between 0 and 1)
`significance_colors`	a named vector with two colors, one for non-significant features named 'FALSE' and one for significant features named 'TRUE'. The default is 'c('FALSE' = 'black', 'TRUE' = 'red')'.
`opacity`	refers to the opacity of the dots in the plot. Values of opacity range from 0 to 1, with lower values corresponding to more transparent colors. The default is `opacity = 0.5`.

Value

A ggplot object with the volcano plot.

Examples

```
# Load example data
# The data are a Feature object containing
# a SummarizedExperiment named "peptide" with MaxQuant peptide intensities
# The data are a subset of spike-in the human-ecoli study
# The variable condition in the colData of the Feature object
# contains information on the spike in condition a-e (from low to high)
data(pe)

# Aggregate peptide intensities in protein expression values
# Note that the peptide intensities were already normalised!
pe <- aggregateFeatures(pe,
                        i = "peptide",
                        fcol = "Proteins",
                        name = "protein")

# Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)

# Define contrast
design <- model.matrix(~condition, data = colData(pe))
# Assess log2 fold change between reference condition a and condition b
L <- makeContrast(
    contrasts = c("conditionb = 0",
                  "conditionc = 0",
                  "conditionc - conditionb = 0"),
    parameterNames = colnames(design))


pe <- hypothesisTest(pe, i = "protein", L)

# Volcano plots
library(ggplot2)
inference <- msqrobCollect(pe[["protein"]], L)
plotVolcano(inference) + facet_wrap(~contrast)
plotVolcano(rowData(pe[["protein"]])$`conditionc`)
plotVolcano(rowData(pe[["protein"]])$`conditionc - conditionb`)
```

`smallestUniqueGroups` *Smallest unique protein groups*

Description

For a given vector of protein group names, outputs the names of those protein groups for which none of its member proteins is present in a smaller protein group.

Usage

```
smallestUniqueGroups(proteins, split = ";")
```

Arguments

proteins	A vector of characters or factors containing single proteins and/or protein groups (i.e. proteins separated by a separator symbol).
split	The character string that is used to separate the indivudual protein names in each protein group.

Value

A character vector containing the names of the protein groups for which none of its proteins is present in a smaller protein group.

Examples

```
data(pe)
smallestUniqueGroups(rowData(pe[["peptide"]])$Proteins)
```

StatModel-class *The StatModel class for msqrob*

Description

The StatModel class contains a statistical model as applied on a feature.

Models are created by the dedicated user-level functions (msqrob(), mqrobAggregate()) or manually, using the StatModel() constructor. In the former case, each quantitative feature is assigned its statistical model and the models are stored as a variable in a DataFrame object, as illustred in the example below.

Function for constructing a new StatModel object.

Usage

```
## S4 method for signature 'StatModel'
show(object)

StatModel(
  type = "fitError",
  params = list(),
  varPosterior = NA_real_,
  dfPosterior = NA_real_
)
```

Arguments

object	StatModel object
type	default set to fit-error, can be a "lm", "rlm" (robust lm with M estimation), "lmer" (when mixed models or ridge regression is adopted), "quasibinomial" (when peptide counts are fitted)
params	A list containing the parameters of the fitted model
varPosterior	Numeric, posterior variance, default is NA
dfPosterior	Numeric, posterior degrees of freedom, default is NA

Value

A StatModel object

Slots

`type` `character(1)` defining type of the used model. Default is `"fitError"`, i.e. a error model. Other include `"lm"`, `"rlm"`, ...

`params` A `list()` containing information of the used model.

`varPosterior` `numeric()` of posterior variance.

`dfPosterior` `numeric()` of posterior degrees of freedom.

Author(s)

Oliver M. Crook, Laurent Gatto, Lieven Clement

Examples

```
## A fully specified dummy model
myModel <- StatModel(
    type = "rlm",
    params = list(x = 3, y = 7, b = 4),
    varPosterior = c(0.1, 0.2, 0.3),
    dfPosterior = c(6, 7, 8)
)
myModel



## A collection of models stored as a variable in a DataFrame
mod1 <- StatModel(type = "rlm")
mod2 <- StatModel(type = "lm")
df <- DataFrame(x = 1:2)
df$mods <- c(mod1, mod2)
df
# TODO
```

topFeatures *Toplist of DE proteins, peptides or features*

Description

Summary table of the differentially expressed Features

Usage

```
topFeatures(models, contrast, adjust.method = "BH", sort = TRUE, alpha = 1)
```

Arguments

models	A list with elements of the class StatModel that are estimated using the msqrob function
contrast	numeric (matrix)vector specifying one contrast of the linear model coefficients to be tested equal to zero. The (row)names of the vector should be equal to the names of parameters of the model.
adjust.method	character specifying the method to adjust the p-values for multiple testing. Options, in increasing conservatism, include '"none"', '"BH"', '"BY"' and '"holm"'. See 'p.adjust' for the complete list of options. Default is "BH" the Benjamini-Hochberg method to controle the False Discovery Rate (FDR).
sort	boolean(1) to indicate if the features have to be sorted according to statistical significance.
alpha	numeric specifying the cutoff value for adjusted p-values. Only features with lower p-values are listed.

Value

A dataframe with log2 fold changes (logFC), standard errors (se), degrees of freedom of the test (df), t-test statistic (t), p-values (pval) and adjusted pvalues (adjPval) using the specified adjust.method in the p.adjust function of the stats package.

Author(s)

Lieven Clement

Examples

```
data(pe)

# Aggregate peptide intensities in protein expression values
pe <- aggregateFeatures(pe, i = "peptide", fcol = "Proteins", name = "protein")

# Fit msqrob model
pe <- msqrob(pe, i = "protein", formula = ~condition)
```

```
# Define contrast
getCoef(rowData(pe[["protein"]])$msqrobModels[[1]])

# Assess log2 fold change between condition c and condition b:
L <- makeContrast("conditionc - conditionb=0", c("conditionb", "conditionc"))
topDeProteins <- topFeatures(rowData(pe[["protein"]])$msqrobModels, L)
```

Index

∗ **datasets**
    pe, 32
.StatModel *(StatModel-class)*, 35
.computeNfLogMedian, 3
.computeNfLogMedianOfRatios, 3

createPairwiseContrasts, 4

data *(pe)*, 32

getCoef *(getModel,StatModel-method)*, 6
getCoef,StatModel-method
        (getModel,StatModel-method), 6
getContrast
        (getContrast,StatModel-method), 5
getContrast,StatModel-method, 5
getDF *(getModel,StatModel-method)*, 6
getDF,StatModel-method
        (getModel,StatModel-method), 6
getDfPosterior
        (getModel,StatModel-method), 6
getDfPosterior,StatModel-method
        (getModel,StatModel-method), 6
getFitMethod
        (getModel,StatModel-method), 6
getFitMethod,StatModel-method
        (getModel,StatModel-method), 6
getModel *(getModel,StatModel-method)*, 6
getModel,StatModel-method, 6
getSigma *(getModel,StatModel-method)*, 6
getSigma,StatModel-method
        (getModel,StatModel-method), 6
getSigmaPosterior
        (getModel,StatModel-method), 6
getSigmaPosterior,StatModel-method
        (getModel,StatModel-method), 6
getVar *(getModel,StatModel-method)*, 6
getVar,StatModel-method
        (getModel,StatModel-method), 6
getVarPosterior
        (getModel,StatModel-method), 6
getVarPosterior,StatModel-method
        (getModel,StatModel-method), 6
getVcovUnscaled
        (getModel,StatModel-method), 6
getVcovUnscaled,StatModel-method
        (getModel,StatModel-method), 6

hypothesisTest
        (hypothesisTest,SummarizedExperiment-method), 8
hypothesisTest,QFeatures-method
        (hypothesisTest,SummarizedExperiment-method), 8
hypothesisTest,SummarizedExperiment-method, 8
hypothesisTestHurdle
        (hypothesisTest,SummarizedExperiment-method), 8
hypothesisTestHurdle,QFeatures-method
        (hypothesisTest,SummarizedExperiment-method), 8
hypothesisTestHurdle,SummarizedExperiment-method
        (hypothesisTest,SummarizedExperiment-method), 8

makeContrast, 11
msqrob, *5*, *37*
msqrob
        (msqrob,SummarizedExperiment-method), 12
msqrob,QFeatures-method
        (msqrob,SummarizedExperiment-method), 12
msqrob,SummarizedExperiment-method, 12
msqrobAggregate
        (msqrobAggregate,SummarizedExperiment-method), 15

msqrobAggregate,QFeatures-method
(msqrobAggregate,SummarizedExperiment-method), 15
msqrobAggregate,SummarizedExperiment-method, 15
msqrobCollect, 17
msqrobGlm, 19
msqrobHurdle
(msqrobHurdle,SummarizedExperiment-method), 20
msqrobHurdle,QFeatures-method
(msqrobHurdle,SummarizedExperiment-method), 20
msqrobHurdle,SummarizedExperiment-method, 20
msqrobLm, 22
msqrobLmer, 23
msqrobQB
(msqrobQB,SummarizedExperiment-method), 25
msqrobQB,QFeatures-method
(msqrobQB,SummarizedExperiment-method), 25
msqrobQB,SummarizedExperiment-method, 25

nfLogMedian
(nfLogMedian,matrix-method), 27
nfLogMedian,matrix-method, 27
nfLogMedian,QFeatures-method
(nfLogMedian,matrix-method), 27
nfLogMedian,SummarizedExperiment-method
(nfLogMedian,matrix-method), 27
nfLogMedianOfRatios
(nfLogMedianOfRatios,matrix-method), 29
nfLogMedianOfRatios,matrix-method, 29
nfLogMedianOfRatios,QFeatures-method
(nfLogMedianOfRatios,matrix-method), 29
nfLogMedianOfRatios,SummarizedExperiment-method
(nfLogMedianOfRatios,matrix-method), 29

pe, 32
plotVolcano, 33

show,StatModel-method
(StatModel-class), 35
smallestUniqueGroups, 34
StatModel (StatModel-class), 35
StatModel-class, 35
StatModel-method
(getContrast,StatModel-method), 5
statModelAccessors
(getModel,StatModel-method), 6
statModelMethods
(getContrast,StatModel-method), 5
topFeatures, 37

varContrast
(getContrast,StatModel-method), 5
varContrast,StatModel-method
(getContrast,StatModel-method), 5