Package ‘MBQN’

February 2, 2024

Version 2.14.0
Title Mean/Median-balanced quantile normalization
Description Modified quantile normalization for omics or other matrix-like
data distorted in location and scale.
License GPL-3 + file LICENSE
Depends R (>= 3.6)
Imports stats, graphics, utils, limma (>= 3.30.13),
SummarizedExperiment (>= 1.10.0), preprocessCore (>= 1.36.0),
BiocFileCache, rappdirs, xml2, RCurl, ggplot2, PairedData,
rmarkdown
Suggests knitr
biocViews Normalization, Preprocessing, Proteomics, Software
Encoding UTF-8
LazyData true
URL https://github.com/arianeschad/mbqn
BugReports https://github.com/arianeschad/MBQN/issues
NeedsCompilation no
RoxygenNote 7.1.2
Roxygen list(markdown = TRUE)
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/MBQN
git_branch RELEASE_3_18
git_last_commit 33f2515
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-02-02
Author Eva Brombacher [aut, cre]
   <https://orcid.org/0000-0002-5488-0985>),
   Clemens Kreutz [aut, ctb]
   <https://orcid.org/0000-0002-8796-5766>),
   Ariane Schad [aut, ctb]
   <https://orcid.org/0000-0002-1921-8902>)
Maintainer Eva Brombacher <brombach@imbi.uni-freiburg.de>
Missing value pattern dataset

Description
An exemplary matrix of a missing value (MV) pattern extracted from LFQ intensities of the proteinGroups.txt dataset from PXD001584 [1].

Usage
example_NApattern

Format
A matrix of zero and ones with 1264 rows and 18 columns; 0 means MV, 1 means no MV.

Author(s)
Ariane Schad

Source
https://www.ebi.ac.uk/pride/archive/projects/PXD001584

References
getKminmax

**Get the k largest/smallest elements**

**Description**

Extract the k largest or smallest values and their indices for each column of a matrix.

**Usage**

```
getKminmax(x, k, flag = "max")
```

**Arguments**

- `x`: a data matrix or data frame.
- `k`: an integer specifying the number of extreme values. Must be <= `nrows(x)`.
- `flag`: use "min" or "max" (default) to select smallest or largest elements.

**Details**

Order the values of each column of `x` and determine the k smallest (`flag = "min"`) or largest (`flag = "max"`) values and their indices. NA's in the data are ignored.

**Value**

List with elements:

- `ik`: indices of ordered extreme values
- `minmax`: ordered extreme values.

**Author(s)**

Ariane Schad

**References**


**Examples**

```r
# Create a data matrix
x <- matrix(c(5,2,3,NA,4,1,4,2,3,4,6,NA,1,3,1),ncol=3)
# Get indices of the 5 largest values in each column
getKminmax(x, k = 5, "max")
```
getPvalue  

Calculates Pitman-Morgan variance test on two matrices

Description

Calculates Pitman-Morgan variance test on two matrices

Usage

getPvalue(mtx1, mtx2)

Arguments

mtx1  
Matrix with samples in columns and features in rows

mtx2  
Matrix with samples in columns and features in rows

Value

Data frame with p values and statistics

Examples

set.seed(30)
n <- 20
m <- 20
mtx1 <- matrix(rnorm(m * n), m, n)
mtx2 <- mbqn(mtx1, FUN = "mean")
getPvalue(mtx1, mtx2)

mbqn  

Mean/Median-balanced quantile normalization

Description

Modified quantile-normalization (QN) of a matrix, e.g., intensity values from omics data or other data sorted in columns. The modification prevents systematic flattening of features (rows) which are rank invariant (RI) or nearly rank invariant (NRI) across columns, for example features that populate mainly the tails of the intensity distribution or features that separate in intensity.
mbqn

Usage

mbqn(
  x,
  FUN = "mean",
  na.rm = TRUE,
  method = "limma",
  offsetmatrix = FALSE,
  verbose = FALSE
)

Arguments

x         a data matrix, where rows represent features, e.g. of protein abundance, and columns represent groups or samples, e.g. replicates, treatments, or conditions.
FUN       a function like mean, median (default), a user defined function, or a numeric vector of weights with length nrow(x) to balance each feature across samples. Functions can be parsed also as characters. If FUN = NULL, features are not balanced, i.e. normal QN is used.
na.rm     logical indicating to omit NAs in the computation of feature mean.
method    character specifying function for computation of quantile normalization; "limma" (default) for normalizeQuantiles() from the limma package or "preprocessCore" for normalize.quantiles() from the preprocessCore package.
offsetmatrix logical indicating if offset matrix should be used instead of offset vector specifying offset for each row
verbose   logical indicating to print messages.

Details

Balance each matrix row by substracting its feature offset computed with FUN, e.g. the median; apply quantile-normalization and add the feature means to the normalized matrix. For further details see [4]. For quantile normalization with the "limma" package see [1,2] and for the preProcessCore package see [3].

Value

Normalized matrix

Author(s)

Ariane Schad

References

mbqnBoxplot

Combined box plot and line plot

Description

Create a box-and-whisker plot of a data matrix and plot selected features and/or additional user-defined data on top of it.

Usage

mbqnBoxplot(mtx, irow = NULL, vals = NULL, add.leg = TRUE, ...)

Arguments

- mtx: a matrix or data frame.
- irow: index or vector of row indices of matrix features to plot on top of the boxplot.
- vals: numeric, array, matrix, or data frame of features with length ncol(mtx) to plot on top of the boxplot.
- add.leg: add legend to plot.
- ...: additional arguments passed to the plot functions, e.g. xlab, ylab, main, xlim, type, las.

Details

This function calls graphics::boxplot. Groups are represent by matrix columns. Selected rows/features or user-defined arrays are plot on top of the box plot. Missing values are ignored.
Value

Figure.

Author(s)

Ariane Schad

References


Examples

```r
## Create boxplot of quantile normalized data matrix and plot
## feature from median balanced quantile normalization on top of it.
X <- matrix(c(5,2,3,NA,4,1,4,2,3,4,6,NA,1,3,1),ncol=3) # Create data matrix
# Quantile normalization
qn.dat <- mbqn(x=X,FUN = NULL ,na.rm = TRUE)
# Median balanced quantile normalization
mbqn.dat <- mbqn(x=X,FUN = median ,na.rm = TRUE)
## Create boxplot:
plot.new()
mbqnBoxplot(qn.dat,irow = 1, vals = mbqn.dat[1,], type = "b")
```

---

`mbqnGetIntersect`  
*Helper function for mbqnGetThreshold*

Description

Helper function for mbqnGetThreshold

Usage

`mbqnGetIntersect(combined_qn, combined_mbqn, threshold, plot = TRUE)`

Arguments

- `combined_qn`: Data frame containing RI, p value and statistic calculated for QN
- `combined_mbqn`: Data frame containing RI, p value and statistic calculated for MBQN
- `threshold`: Significance threshold for p value of Pitman-Morgan variance test
- `plot`: Boolean values if logistic regression curves that are used to calculate intersection point should be plotted

Value

threshold value
Identify rank invariant (RI) and nearly rank invariant (NRI) features

**Description**

Compute the rank frequency of each feature of a matrix and identify NRI/RI features.

**Usage**

```r
mbqnGetNRIfeatures(x, low_thr = 0.5, method = NULL, verbose = TRUE)
```

**Arguments**

- `x`: a data matrix. Rows represent features, e.g. protein abundances; columns represent samples.
- `low_thr`: a value between [0 1]. Features with RI frequency \( \geq \text{low\_thr} \) are considered as NRI/RI; default 0.5.
- `method`: character specifying function for computation of quantile normalization; "limma" (default) for `normalizeQuantiles()` from the limma package or "preprocessCore" for `normalize.quantiles()` from the preprocessCore package.
- `verbose`: logical indicating to print messages.

**Details**

Quantile normalize the data matrix and sort ranks. Determine the maximum frequency of equal rank across all columns for each feature. Features with maximum frequency above the user-defined threshold are declared as nearly rank invariant.

**Value**

A list with elements:

- `p`: a matrix with the rank invariance frequencies `ri.freq` and the sample coverage `sample.coverage` for all detected RI/NRI features
- `max_p`: maximum rank invariance frequency in percent
- `ip`: index of feature with maximum rank invariance frequency
- `nri`: table of the rank invariance frequencies in percent for each NRI/RI feature
- `var0_feature`: indices of features with zero sample variance after QN
- `low_thr`: threshold used for NRI/RI detection from RI frequency.

**Author(s)**

Ariane Schad
mbqnGetThreshold

References

See Also
mbqnPlotRI() for visualization of detected NRI/RI features.

Examples
## Check data matrix for RI and NRI features
set.seed(1234)
x <- mbqnSimuData("omics.dep")
RI <- mbqnGetNRIfeatures(x, low_thr = 0.5, verbose = FALSE)
mbqnPlotRI(RI)

mbqnGetThreshold      Calculates the rank invariance threshold from which on MBQN should be used instead of 'classical' QN

Description
Calculates the rank invariance threshold from which on MBQN should be used instead of 'classical' QN

Usage
mbqnGetThreshold(mtx, meanMedian = "mean", plot = TRUE)

Arguments
- mtx: Matrix with samples in columns and features in rows
- meanMedian: Offset function for the MBQN calculation
- plot: Boolean values if logistic regression curves that are used to calculate intersection point should be plotted

Value
threshold value

Examples
set.seed(30)
n <- 20
m <- 20
mtx <- matrix(rnorm(m * n), m, n)
mbqnGetThreshold(mtx)
**mbqnNRI**  
*Selective mean/median-balanced quantile normalization*

**Description**

Quantile normalization of a data matrix where rank invariant (RI)/nearly rank invariant (NRI) rows/features or other user-selected rows are normalized by the mean/median-balanced quantile normalization.

**Usage**

```r
mbqnNRI(
  x,
  FUN = "mean",
  na.rm = TRUE,
  method = NULL,
  low_thr = 0.5,
  index = NULL,
  offsetmatrix = FALSE,
  verbose = TRUE
)
```

**Arguments**

- `x`: a data matrix, where rows represent features, e.g. of protein abundance, and columns represent groups or samples, e.g. replicates, treatments, or conditions.
- `FUN`: a function like mean, median (default), a user defined function, or a numeric vector of weights with length `nrow(x)` to balance each feature across samples. Functions can be parsed also as characters. If `FUN = NULL`, features are not balanced, i.e. normal QN is used.
- `na.rm`: logical indicating to omit NAs in the computation of feature mean.
- `method`: character specifying function for computation of quantile normalization; "limma" (default) for `normalizeQuantiles()` from the limma package or "preprocessCore" for `normalize.quantiles()` from the preprocessCore package.
- `low_thr`: a value between [0 1]. Features with RI frequency >= `low_thr` are considered as NRI/RI; default 0.5.
- `index`: an integer or a vector integers specifying the indices of selected rows.
- `offsetmatrix`: logical indicating if offset matrix should be used instead of offset vector specifying offset for each row.
- `verbose`: logical indicating to print messages.

**Details**

Selected rows and/or rows with rank invariance frequency >= threshold are normalized with the mean/median-balanced quantile normalization. Remaining rows are quantile normalized without mean balancing.
Value

Normalized matrix.

Author(s)

Ariane Schad

References


See Also

mbqn(), mbqnGetNRIfeatures().

Examples

## Quantile normalize a data matrix where
## nearly rank invariant (NRI) features are balanced
X <- matrix(c(5,2,3,NA,4,1,4,2,3,4,6,NA,1,3,1),ncol=3)
mbqnNRI(X, median, low_thr = 0.5) # Balance NRI features selected by threshold
mbqnNRI(X, median, index = c(1,2)) # Balance selected features

---

**mbqnPlotAll**

Plot RI/NRI feature frequencies and normalized/unnormalized features

Description

Check data matrix for rank invariant (RI) and nearly rank invariant (NRI) features/rows across samples and visualize result for different normalizations.

Usage

```
mbqnPlotAll(
  x,
  FUN = NULL,
  low_thr = 0.5,
  show_nri_only = FALSE,
  verbose = TRUE,
  ...)
```
Arguments

- **x**: a data matrix. Rows represent features, e.g. protein abundances; columns represent samples.
- **FUN**: a function like mean, median (default), a user defined function, or a numeric vector of weights with length `nrow(x)` to balance each feature across samples. Functions can be parsed also as characters. If `FUN = NULL`, features are not balanced, i.e. normal QN is used.
- **low_thr**: a value between [0 1]. Features with RI frequency >= `low_thr` are considered as NRI/RI; default 0.5.
- **show_nri_only**: logical indicating to display only the RI/NRI detection graph.
- **verbose**: logical indicating to print messages.
- **...**: additional plot arguments passed to `mbqnBoxplot`, and `mbqnPlotRI`.

Details

Rank data and check if lower and upper intensity tails are dominated by few features. Apply quantile normalization without and with mean-balancing and check the standard deviation of normalized features located in the tails.

Value

A set of figures that display the detected RI/NRI features and a list with elements:

- **p**: a matrix with the rank invariance frequencies `ri.freq` and the sample coverage `sample.coverage` for all detected RI/NRI features.
- **max_p**: maximum rank invariance frequency in percent
- **ip**: index of feature with maximum rank invariance frequency
- **nri**: table of the rank invariance frequencies in percent for each NRI/RI feature
- **var0_feature**: indices of features with zero sample variance after QN.

Author(s)

Ariane Schad

References


See Also

`mbqnPlotRI()` and `mbqnBoxplot()` for the generation of figures, and `mbqn()` for normalization.

Examples

```r
## Check data matrix for RI and NRI features
X <- matrix(c(5,2,3,NA,4,1,4,2,3,4,6,NA,1,3,1),ncol=3)
mbqnPlotAll(X, mean, low_thr = 0.5)
```
Description

Plot rank invariance frequency and feature coverage of detected RI and NRI features

Usage

```r
mbqnPlotRI(obj, verbose = FALSE, ...)
```

Arguments

- `obj`: list object of RI frequencies from `mbqnGetNRIfeatures()`.
- `verbose`: logical indicating to run function quietly.
- `...`: additional arguments (cex, cex.lab, cex.axis, cex.main) passed to the plot function.

Details

Graphical output of the NRI/RI identification results from `mbqnGetNRIfeatures()`. For each detected NRI/RI feature, plot the feature index against the RI frequencies together with the RI frequency detection threshold and print the sample coverage.

Value

Figure

Author(s)

Ariane Schad

References


See Also

`mbqnGetNRIfeatures()` for detection of NRI/RI features.

Examples

```r
## Check data matrix for RI and NRI features
x <- mbqnSimuData("omics.dep")
RI <- mbqnGetNRIfeatures(x, low_thr = 0.5, verbose = FALSE)
mbqnPlotRI(RI)
```
Generate a random data matrix with or without proteomics, log-transformed feature intensity-like properties.

Usage

```r
mbqnSimuData(model = "rand", nrow = NULL, ncol = NULL, show.fig = FALSE)
```

Arguments

- `model` character indicating one of the three different type of models: "rand" (default) a Gaussian random matrix of size nrow x ncol (default 1000 x 10), "omics" a Gaussian random matrix of size 1264 x 18 that mimics intensity profiles and missing values as present in real data, and "omics.dep" is the same as "omics" but with an additional single, differentially expressed RI feature.
- `nrow` number of rows of data matrix (only for `model = "rand"`).
- `ncol` number of columns of data matrix (only for `model = "rand"`).
- `show.fig` logical indicating whether data properties are plot to figure (only for `model = "omics"` and `model = "omics.dep"`).

Details

For model "rand", each matrix element is drawn from a standard normal distribution $N(0, 1)$. For model "omics", the matrix elements of each row are drawn from a Gaussian distribution $N(\mu_i, \sigma_i^2)$ where the mean and standard deviation itself are drawn Gaussian distributions, i.e. $\sigma_i ~ N(0, 0.0625)$ and $\mu_i ~ N(28, 4)$. About 35\% to the missing value pattern present in real protein LFQ intensities. For model "omics.dep", a single differentially expressed RI feature is stacked on top of the matrix from model "omics".

Value

matrix of size nrow x ncol.

Author(s)

Ariane Schad

References


See Also

`example_NApattern()` for description of missing value pattern.
mbqnSimuDistortion

Examples

mbqnSimuData(model = "rand")
mbqnSimuData(model = "rand", 2000,6)
set.seed(1234)
mbqnSimuData(model = "omics")
set.seed(1111)
mbqnSimuData(model = "omics.dep")

---

mbqnSimuDistortion  *Perturbation of sample mean and scale*

Description

mbqnSimuDistortion adds a random perturbation of mean and scale to each column of a matrix.

Usage

mbqnSimuDistortion(x, s.mean = 0.05, s.scale = 0.01)

Arguments

x  
a matrix or data frame.

s.mean  
scatter of relative change of mean.

s.scale  
scatter of relative change in scale, i.e. 0.01 corresponds to 1 percent.

Details

Shift and scale the sample mean and standard deviation of a matrix. The perturbation of center and scale relative to mean and standard deviation of each sample are drawn from a Gaussian distribution $N(0, \sigma^2)$ with $\sigma_{mean}=s\text{.mean}$ and $\sigma_{scale}=s\text{.scale}$, respectively.

Value

List with:

x.mod  
perturbed matrix

mx.offset  
numeric array of shifts of the sample means

mx.scale  
numeric array of relative scales of the sample standard deviations.

Author(s)

Ariane Schad

References

See Also

`mbqnSimuData()` for data generation.

Examples

```r
set.seed(1234)
x <- mbqnSimuData("omics.dep")
df <- mbqnSimuDistorion(x)
```

---

oneSidedTest  
Recalculate p value from two-sided to one-sided

Description

Recalculate p value from two-sided to one-sided

Usage

```r
oneSidedTest(sign_value, z_value)
```

Arguments

- `sign_value`  
P value from two-sided significance test
- `z_value`  
Z value from two-sided significance test

Value

P value from one sided significance test

---

truncateDecimals  
Truncate float to defined number of decimal values

Description

Truncate float to defined number of decimal values

Usage

```r
truncateDecimals(x, digits = 2)
```

Arguments

- `x`  
float
- `digits`  
Number of decimal values
`truncateDecimals`

**Value**

Truncated number

**Examples**

```r
x <- 2.567836
truncateDecimals(x, 3)
```
Index

* datasets
  example_NApattern, 2

* data
  example_NApattern, 2

* example
  mbqnBoxplot, 6

example_NApattern, 2
eample_NApattern(), 14

getKminmax, 3
getPvalue, 4

mbqn, 4
mbqn(), 11, 12
mbqnBoxplot, 6
mbqnBoxplot(), 12
mbqnGetIntersect, 7
mbqnGetNRIfeatures, 8
mbqnGetNRIfeatures(), 6, 11, 13
mbqnGetThreshold, 9
mbqnNRI, 10
mbqnNRI(), 6
mbqnPlotAll, 11
mbqnPlotRI, 13
mbqnPlotRI(), 9, 12
mbqnSimuData, 14
mbqnSimuData(), 16
mbqnSimuDistortion, 15

oneSidedTest, 16

truncateDecimals, 16