Package 'NormalyzerDE'

May 26, 2022

Title Evaluation of normalization methods and calculation of differential expression analysis statistics

Version 1.15.0

Author Jakob Willforss

Description NormalyzerDE provides screening of normalization methods for LC-MS based expression data. It calculates a range of normalized matrices using both existing approaches and a novel time-segmented approach, calculates performance measures and generates an evaluation report. Furthermore, it provides an easy utility for Limma- or ANOVA- based differential expression analysis.

Imports vsn, preprocessCore, limma, MASS, ape, car, ggplot2, methods, Biobase, RcmdrMisc, raster, utils, stats, SummarizedExperiment, matrixStats, ggforce

Suggests knitr, testthat, rmarkdown, roxygen2, hexbin, BiocStyle

VignetteBuilder knitr

biocViews Normalization, MultipleComparison, Visualization, Bayesian, Proteomics, Metabolomics, DifferentialExpression

License Artistic-2.0

Encoding UTF-8

RoxygenNote 7.1.0

URL https://github.com/ComputationalProteomics/NormalyzerDE

Depends R (>= 3.6)

git_url https://git.bioconductor.org/packages/NormalyzerDE

git_branch master

git_last_commit 4c3d0e2

git_last_commit_date 2022-04-26

Date/Publication 2022-05-26

Maintainer Jakob Willforss <jakob.willforss@hotmail.com>

R topics documented:

`analyzeNormalizations` *Calculate measures for normalization results*

Description

This function prepares an NormalyzerEvaluationResults object containing the evaluation measures CV (coefficient of variance), MAD (median absolute deviation), average variance, significance measures (ANOVA between condition groups) and correlation between replicates.

Usage

```
analyzeNormalizations(nr, categoricalAnova = FALSE)
```

Arguments

nr	Normalyzer results object with calculated results.
categoricalAnova	Whether categorical or numerical (ordered) ANOVA should be calculated.

Value

Normalyzer results with attached evaluation results object.

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
normResults <- normMethods(normObj)
normResultsWithEval <- analyzeNormalizations(normResults)
```

---

calculateContrasts

Performs statistical comparisons between the supplied conditions. It uses the design matrix and data matrix in the supplied NormalyzerStatistics object. A column is supplied specifying which of the columns in the design matrix that is used for deciding the sample groups. The comparisons vector specifies which pairwise comparisons between condition levels that are to be calculated.

Description

Optionally, a batch column can be specified allowing compensation for covariate variation in the statistical model. This is only compatible with a Limma-based statistical analysis.

Usage

```
calculateContrasts(
  nst,
  comparisons,
  condCol,
  batchCol = NULL,
  splitter = "-",
  type = "limma",
  leastRepCount = 1
)

## S4 method for signature 'NormalyzerStatistics'
calculateContrasts(
  nst,
  comparisons,
  condCol,
  batchCol = NULL,
```

```
  splitter = "-",
  type = "limma",
  leastRepCount = 1
)
```

Arguments

nst	Results evaluation object.
comparisons	String with comparisons for contrasts.
condCol	Column name in design matrix containing condition information.
batchCol	Column name in design matrix containing batch information.
splitter	Character dividing contrast conditions.
type	Type of statistical test (Limma or welch).
leastRepCount	Least replicates in each group to be retained for contrast calculations

Value

nst Statistics object with statistical measures calculated

Examples

```
data(example_stat_summarized_experiment)
nst <- NormalyzerStatistics(example_stat_summarized_experiment)
results <- calculateContrasts(nst, c("1-2", "2-3"), "group")
resultsBatch <- calculateContrasts(nst, c("1-2", "2-3"), "group", batchCol="batch")
```

---

generateAnnotatedMatrix

Generate an annotated data frame from statistics object

Description

Extracts key values (p-value, adjusted p-value, log2-fold change and average expression values) from an NormalyzerStatistics instance and appends these to the annotation- and data-matrices

Usage

```
generateAnnotatedMatrix(nst, prefixSep = "_", compLabels = NULL)
```

Arguments

nst	NormalyzerDE statistics object.
prefixSep	Character string for separating the prefix names from the statistics suffix
compLabels	Vector containing strings to use as prefix for statistical comparisons

Value

outDf Annotated statistics matrix

Examples

```
data(example_stat_summarized_experiment)
statObj <- NormalyzerStatistics(example_stat_summarized_experiment)
statObj <- calculateContrasts(statObj, comparisons=c("1-2", "2-3"), condCol="group", type="limma")
annotDf <- generateAnnotatedMatrix(statObj)
```

`generatePlots` *Generates a number of visualizations for the performance measures calculated for the normalized matrices. These contain both general measures and direct comparisons for different normalization approaches.*

Description

They include:

Usage

```
generatePlots(nr, jobdir, plotRows = 3, plotCols = 4, writeAsPngs = FALSE)
```

Arguments

`nr`	Normalyzer results object.
`jobdir`	Path to output directory for run.
`plotRows`	Number of plot rows.
`plotCols`	Number of plot columns.
`writeAsPngs`	Output the report as PNG-plots instead of a single PDF

Details

"Total intensity" Barplot showing the summed intensity in each sample for thelog2-transformed data

"Total missing" Barplot showing the number of missing values found in each sample for the log2-tranformed data

Log2-MDS plot: MDS plot where data is reduced to two dimensions allowing inspection of the main global changes in the data

PCV - Intragroup: Mean of intragroup CV of all replicate groups

PMAD - Intragroup: Mean of intragroup median absolute deviation across replicate groups

PEV - Intragroup: Mean of intragroup pooled estimate of variance across the replicate groups

Relative PCV, PMAD and PEV compared to log2: The results from PCV, PMAD and PEV from all normalized data compared to the log2 data

Stable variables plot: 5 analysis of log2 transformed data. Thereafter, global CV of these variables is estimated from different normalized datasets. A plot of global CV of the stable variables from all datsets on the y-axis and PCV-compared to log2 on the x-axis is generated.

CV vs Raw Intensity plots: For the first replicate group in each of the normalized dataset, a plot of PCV of each variable compared to the average intensity of the variable in the replicate group is plotted.

MA plots: Plotted using the plotMA function of the limma package. The first sample in each dataset is plotted against the average of the replicate group that sample belong to.

Scatterplots: The first two samples from each dataset are plotted.

Q-Q plots: QQ-plots are plotted for the first sample in each normalized dataset.

Boxplots: Boxplots for all samples are plotted and colored according to the replicate grouping.

Relative Log Expression (RLE) plots: Relative log expression value plots. Ratio between the expression of the variable and the median expression of this variable across all samples. The samples should be aligned around zero. Any deviation would indicate discrepancies in the data.

Density plots: Density distributions for each sample using the density function. Can capture outliers (if single densities lies far from the others) and see if there is batch effects in the dataset (if for instance there is two clear collections of lines in the data).

MDS plots Multidimensional scaling plot using the cmdscale() function from the stats package. Is often able to show whether replicates group together, and whether there are any clear outliers in the data.

MeanSDplots Displays the standard deviation values against values ordered according to mean. If no dependency on mean is present (as is desired) a flat red line is shown.

Pearson and Spearman correlation Mean of intragroup Pearson and Spearman correlation values for each method.

Dendograms Generated using the hclust function. Data is centered and scaled prior to analysis. Coloring of replicates is done using as.phylo from the ape package.

P-value histograms Histogram plots of p-values after calculating an ANOVA between different condition groups. If no effect is present in the data a flat distribution is expected. If an effect is present a flat distribution is still expected, but with a sharp peak close to zero. If other effects are present it might indicate that the data doesn't support the assumptions of ANOVA, for instance if there are batch effects present in the data.

Value

None

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
normResults <- normMethods(normObj)
normResultsWithEval <- analyzeNormalizations(normResults)
outputDir <- tempdir()
generatePlots(normResultsWithEval, outputDir)
```

`generateStatsReport` *Generate full output report plot document. Plots p-value histograms for each contrast in the NormalyzerStatistics instance and writes these to a PDF report.*

Description

Generate full output report plot document. Plots p-value histograms for each contrast in the NormalyzerStatistics instance and writes these to a PDF report.

Usage

```
generateStatsReport(
  nst,
  jobName,
  jobDir,
  sigThres = 0.1,
  sigThresType = "fdr",
  log2FoldThres = 0,
  plotRows = 3,
  plotCols = 4,
  writeAsPngs = FALSE
)
```

Arguments

`nst`	NormalyzerDE statistics object.
`jobName`	Name of processing run.
`jobDir`	Path to output directory.
`sigThres`	Significance threshold for indicating as significant
`sigThresType`	Type of significance threshold (FDR or p)
`log2FoldThres`	log2 fold-change required for being counted as significant
`plotRows`	Number of plot rows.
`plotCols`	Number of plot columns.
`writeAsPngs`	Output the report as separate PNG files instead of a single PDF file

Value

None

Examples

```
data(example_stat_summarized_experiment)
statObj <- NormalyzerStatistics(example_stat_summarized_experiment)
statObj <- calculateContrasts(statObj, comparisons=c("1-2", "2-3"),
  condCol="group", type="limma")
outputDir <- tempdir()
generateStatsReport(statObj, "jobName", outputDir)
```

---

getRTNormalizedMatrix *Perform RT-segmented normalization by performing the supplied normalization over retention-time sliced data*

Description

The function orders the retention times and steps through them using the supplied step size (in minutes). If smaller than a fixed lower boundary the window is expanded to ensure a minimum amount of data in each normalization step. An offset can be specified which can be used to perform multiple RT-segmentations with partial overlapping windows.

Usage

```
getRTNormalizedMatrix(
  rawMatrix,
  retentionTimes,
  normMethod,
  stepSizeMinutes = 1,
  windowMinCount = 100,
  offset = 0,
  noLogTransform = FALSE
)
```

Arguments

`rawMatrix`	Target matrix to be normalized
`retentionTimes`	Vector of retention times corresponding to rawMatrix
`normMethod`	The normalization method to apply to the time windows
`stepSizeMinutes`	Size of windows to be normalized
`windowMinCount`	Minimum number of values for window to not be expanded.
`offset`	Whether time window should shifted half step size
`noLogTransform`	Don't log-transform the data

Value

Normalized matrix

Examples

```
data(example_data_small)
data(example_design_small)
data(example_data_only_values)
dataMat <- example_data_only_values
retentionTimes <- as.numeric(example_data[, "Average.RT"])
performCyclicLoessNormalization <- function(rawMatrix) {
    log2Matrix <- log2(rawMatrix)
    normMatrix <- limma::normalizeCyclicLoess(log2Matrix, method="fast")
    colnames(normMatrix) <- colnames(rawMatrix)
    normMatrix
}
rtNormMat <- getRTNormalizedMatrix(dataMat, retentionTimes,
performCyclicLoessNormalization, stepSizeMinutes=1, windowMinCount=100)
```

`getSmoothedRTNormalizedMatrix`

Generate multiple RT time-window normalized matrices where one is shifted. Merge them using a specified method (mean or median) and return the result.

Description

Uses the function getRTNormalizedMatrix to generate multiple normalized matrices which are shifted respective to each other and finally merged into a single matrix. This could potentially reduce effect of fluctuations within individual windows.

Usage

```
getSmoothedRTNormalizedMatrix(
  rawMatrix,
  retentionTimes,
  normMethod,
  stepSizeMinutes,
  windowShifts = 2,
  windowMinCount = 100,
  mergeMethod = "mean",
  noLogTransform = FALSE
)
```

Arguments

`rawMatrix`	Target matrix to be normalized
`retentionTimes`	Vector of retention times corresponding to rawMatrix
`normMethod`	The normalization method to apply to the time windows
`stepSizeMinutes`	Size of windows to be normalized

windowShifts	Number of frame shifts.
windowMinCount	Minimum number of features within window.
mergeMethod	Layer merging approach. Mean or median.
noLogTransform	Don't log transform the input

Value

Normalized matrix

Examples

```
data(example_data_small)
data(example_data_only_values)
data(example_design_small)
retentionTimes <- as.numeric(example_data[, "Average.RT"])
dataMat <- example_data_only_values
performCyclicLoessNormalization <- function(rawMatrix) {
    log2Matrix <- log2(rawMatrix)
    normMatrix <- limma::normalizeCyclicLoess(log2Matrix, method="fast")
    colnames(normMatrix) <- colnames(rawMatrix)
    normMatrix
}
rtNormMat <- getSmoothedRTNormalizedMatrix(dataMat, retentionTimes,
    performCyclicLoessNormalization, stepSizeMinutes=1, windowMinCount=100,
    windowShifts=2, mergeMethod="median")
```

---

getVerifiedNormalyzerObject

Verify that input data is in correct format, and if so, return a generated NormalyzerDE data object from that input data

---

Description

This function performs a number of checks on the input data and provides informative error messages if the data isn't fulfilling the required format. Checks include verifying that the design matrix matches to the data matrix, that the data matrix contains valid numbers and that samples have enough values for analysis

Usage

```
getVerifiedNormalyzerObject(
  jobName,
  summarizedExp,
  threshold = 15,
  omitSamples = FALSE,
  requireReplicates = TRUE,
  quiet = FALSE,
```

```
  noLogTransform = FALSE,
  tinyRunThres = 50
)
```

Arguments

jobName	Name of ongoing run.
summarizedExp	Summarized experiment input object
threshold	Minimum number of features.
omitSamples	Automatically omit invalid samples from analysis.
requireReplicates	Require there to be at least to samples per condition
quiet	Don't print output messages during processing
noLogTransform	Don't log-transform the provided data
tinyRunThres	If less features in run, a limited run is performed

Value

Normalyzer data object representing verified input data.

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
```

---

globalIntensityNormalization

The normalization divides the intensity of each variable in a sample with the sum of intensities of all variables in the sample and multiplies with the median of sum of intensities of all variables in all samples. The normalized data is then log2-transformed.

Description

The normalization divides the intensity of each variable in a sample with the sum of intensities of all variables in the sample and multiplies with the median of sum of intensities of all variables in all samples. The normalized data is then log2-transformed.

Usage

```
globalIntensityNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized and log-transformed matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- globalIntensityNormalization(example_data_only_values)
```

`loadData` *Load raw data into dataframe*

Description

General function which allows specifying different types of input data including "proteios", "maxquantpep" (peptide output from MaxQuant) and "maxquantprot" (protein output from MaxQuant) formats.

Usage

```
loadData(dataPath, inputFormat = "default")
```

Arguments

`dataPath`	File path to design matrix.
`inputFormat`	If input is given in standard NormalyzerDE format, Proteios format or in MaxQuant protein or peptide format

Value

rawData Raw data loaded into data frame

Examples

```
## Not run:
df <- loadData("data.tsv")

## End(Not run)
```

`loadDesign` *Load raw design into dataframe*

Description

Takes a design path, loads the matrix and ensures that the sample column is in character format and that the group column is in factor format.

Usage

```
loadDesign(designPath, sampleCol = "sample", groupCol = "group")
```

Arguments

`designPath`	File path to design matrix.
`sampleCol`	Column name for column containing sample names.
`groupCol`	Column name for column containing condition levels.

Value

designMatrix Design data loaded into data frame

Examples

```
## Not run:
df <- loadDesign("design.tsv")

## End(Not run)
```

`meanNormalization` *Intensity of each variable in a given sample is divided by the mean of sum of intensities of all variables in the sample and then multiplied by the mean of sum of intensities of all variables in all samples. The normalized data is then transformed to log2.*

Description

Intensity of each variable in a given sample is divided by the mean of sum of intensities of all variables in the sample and then multiplied by the mean of sum of intensities of all variables in all samples. The normalized data is then transformed to log2.

Usage

```
meanNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized and log-transformed matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- meanNormalization(example_data_only_values)
```

---

medianNormalization *Intensity of each variable in a given sample is divided by the median of intensities of all variables in the sample and then multiplied by the mean of median of sum of intensities of all variables in all samples. The normalized data is then log2-transformed.*

Description

Intensity of each variable in a given sample is divided by the median of intensities of all variables in the sample and then multiplied by the mean of median of sum of intensities of all variables in all samples. The normalized data is then log2-transformed.

Usage

```
medianNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized and log-transformed matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- medianNormalization(example_data_only_values)
```

normalyzer *NormalyzerDE pipeline entry point*

Description

This function is the main execution point for the normalization part of the NormalyzerDE analysis pipeline. When executed it performs the following steps:

Usage

```
normalyzer(
  jobName,
  designPath = NULL,
  dataPath = NULL,
  experimentObj = NULL,
  outputDir = ".",
  forceAllMethods = FALSE,
  omitLowAbundSamples = FALSE,
  sampleAbundThres = 5,
  tinyRunThres = 50,
  requireReplicates = TRUE,
  normalizeRetentionTime = TRUE,
  plotRows = 3,
  plotCols = 4,
  zeroToNA = FALSE,
  sampleColName = "sample",
  groupColName = "group",
  inputFormat = "default",
  skipAnalysis = FALSE,
  quiet = FALSE,
  noLogTransform = FALSE,
  writeReportAsPngs = FALSE,
  rtStepSizeMinutes = 1,
  rtWindowMinCount = 100,
  rtWindowShifts = 1,
  rtWindowMergeMethod = "mean"
)
```

Arguments

jobName	Give the current run a name.
designPath	Path to file containing design matrix.
dataPath	Specify an output directory for generated files. Defaults to current working directory.
experimentObj	SummarizedExperiment object, can be provided as input as alternative to 'designPath' and 'dataPath'

`outputDir` Directory where results folder is created.

`forceAllMethods`
: Debugging function. Run all normalizations even if they aren't in the recommended range of number of values

`omitLowAbundSamples`
: Automatically remove samples with fewer non-NA values compared to threshold given by sampleAbundThres. Will otherwise stop with error message if such sample is encountered.

`sampleAbundThres`
: Threshold for omitting low-abundant samples. Is by default set to 15.

`tinyRunThres` If total number of features is less than this, a limited run is performed.

`requireReplicates`
: Require multiple samples per condition to pass input validation.

`normalizeRetentionTime`
: Perform normalizations over retention time.

`plotRows` Number of plot-rows in output documentation.

`plotCols` Number of plot-columns in output documentation.

`zeroToNA` Convert zero values to NA.

`sampleColName` Column name in design matrix containing sample IDs.

`groupColName` Column name in design matrix containing condition IDs.

`inputFormat` Type of input format.

`skipAnalysis` Only perform normalization steps.

`quiet` Omit status messages printed during run.

`noLogTransform` Don't log-transform the input.

`writeReportAsPngs`
: Output the evaluation report as PNG files instead of a single PDF

`rtStepSizeMinutes`
: Retention time normalization window size.

`rtWindowMinCount`
: Minimum number of datapoints in each retention-time segment.

`rtWindowShifts` Number of layered retention time normalized windows.

`rtWindowMergeMethod`
: Merge approach for layered retention time windows.

Details

1: Loads the data matrix containing expression values and optional annotations, as well as the design matrix containing the experimental setup 2: Performs input data verification to validate that the data is in correct format. This step captures many common formatting errors. It returns an instance of the NormalyzerDataset class representing the unprocessed data. 3: Calculate a range of normalizations for the dataset. The result is provided as a NormalyzerResults object containing the resulting data matrices from each normalization. 4: Analyze the normalizations and generate performance measures for each of the normalized datasets. This result is provided as a NormalyzerEvaluationResults object. 5: Output the matrices containing the normalized datasets to files. 6: Generate visualizations overviewing the performance measures and write them to a PDF report.

Value

None

Examples

```
## Not run:
data_path <- system.file(package="NormalyzerDE", "extdata", "tiny_data.tsv")
design_path <- system.file(package="NormalyzerDE", "extdata", "tiny_design.tsv")
out_dir <- tempdir()
normalyzer(
    jobName="my_jobname",
    designPath=design_path,
    dataPath=data_path,
    outputDir=out_dir)
normalyzer(
    "my_jobname",
    designMatrix="design.tsv",
    "data.tsv",
    outputDir="path/to/output",
    normalizeRetentionTime=TRUE,
    retentionTimeWindow=2)
normalyzer(
    "my_jobname",
    designMatrix="design.tsv",
    "data.tsv",
    outputDir="path/to/output",
    inputFormat="maxquantprot")

## End(Not run)
```

---

normalyzerDE *NormalyzerDE differential expression*

Description

Performs differential expression analysis on a normalization matrix. This command executes a pipeline processing the data and generates an annotated normalization matrix and a report containing p-value histograms for each of the performed comparisons.

Usage

```
normalyzerDE(
  jobName,
  comparisons,
  designPath = NULL,
  dataPath = NULL,
  experimentObj = NULL,
  outputDir = ".",
  logTrans = FALSE,
```

```
  type = "limma",
  sampleCol = "sample",
  condCol = "group",
  batchCol = NULL,
  techRepCol = NULL,
  leastRepCount = 1,
  quiet = FALSE,
  sigThres = 0.1,
  sigThresType = "fdr",
  log2FoldThres = 0,
  writeReportAsPngs = FALSE
)
```

Arguments

jobName	Name of job
comparisons	Character vector containing target contrasts. If comparing condA with condB, then the vector would be c("condA-condB")
designPath	File path to design matrix
dataPath	File path to normalized matrix
experimentObj	SummarizedExperiment object, can be provided as input as alternative to 'designPath' and 'dataPath'
outputDir	Path to output directory
logTrans	Log transform the input (needed if providing non-logged input)
type	Type of statistical comparison, "limma", "limma_intensity" or "welch", where "limma_intensity" allows the prior to be fit according to intensity rather than using a flat prior
sampleCol	Design matrix column header for column containing sample IDs
condCol	Design matrix column header for column containing sample conditions
batchCol	Provide an optional column for inclusion of possible batch variance in the model
techRepCol	Design matrix column header for column containing technical replicates
leastRepCount	Minimum required replicate count
quiet	Omit status messages printed during run
sigThres	Significance threshold use for illustrating significant hits in diagnostic plots
sigThresType	Type of significance threshold, "fdr" or "p". "fdr" is strongly recommended (Benjamini-Hochberg corrected p-values)
log2FoldThres	Fold-size cutoff for being considered significant in diagnostic plots
writeReportAsPngs	Output report as separate PNG files instead of a single PDF

Details

When executed, it performs the following steps:

1: Read the data and the design matrices into dataframes. 2: Generate an instance of the NormalyzerStatistics class representing the data and their statistical comparisons. 3: Optionally reduce technical replicates in both the data matrix and the design matrix 4: Calculate statistical contrats between supplied groups 5: Generate an annotated version of the original dataframe where columns containing statistical key measures have been added 6: Write the table to file 7: Generate a PDF report displaying p-value histograms for each calculated contrast

Value

None

Examples

```
data_path <- system.file(package="NormalyzerDE", "extdata", "tiny_data.tsv")
design_path <- system.file(package="NormalyzerDE", "extdata", "tiny_design.tsv")
out_dir <- tempdir()
normalyzerDE(
  jobName="my_jobname",
  comparisons=c("4-5"),
  designPath=design_path,
  dataPath=data_path,
  outputDir=out_dir,
  condCol="group")
```

NormalyzerEvaluationResults

Representation of evaluation results by calculating performance measures for an an NormalyzerResults instance

---

Description

Contains the resulting information from the processing which subsequently can be used to generate the quality assessment report.

Usage

```
NormalyzerEvaluationResults(nr)

NormalyzerEvaluationResults(nr)
```

Arguments

nr	NormalyzerResults object

Value

nds Generated NormalyzerEvaluationResults instance

Slots

`avgcvmem` Average coefficient of variance per method

`avgcvmempdiff` Percentage difference of mean coefficient of variance compared to log2-transformed data

`featureCVPerMethod` CV calculated per feature and normalization method.

`avgmadmem` Average median absolute deviation

`avgmadmempdiff` Percentage difference of median absolute deviation compared to log2-transformed data

`avgvarmem` Average variance per method

`avgvarmempdiff` Percentage difference of mean variance compared to log2-transformed data

`lowVarFeaturesCVs` List of 5 for log2-transformed data

`lowVarFeaturesCVsPercDiff` Coefficient of variance for least variable entries

`anovaP` ANOVA calculated p-values

`repCorPear` Within group Pearson correlations

`repCorSpear` Within group Spearman correlations

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
normResults <- normMethods(normObj)
normEval <- NormalyzerEvaluationResults(normResults)
```

NormalyzerResults *Representation of the results from performing normalization over a dataset*

Description

It is linked to a NormalyzerDataset instance representing the raw data which has been processed. After performing evaluation it also links to an instance of NormalyzerEvaluationResults representing the results from the evaluation.

Usage

```
NormalyzerResults(nds)

NormalyzerResults(nds)
```

Arguments

nds	NormalyzerDataset object

Value

nr Prepared NormalyzerResults object

Slots

`normalizations` SummarizedExperiment object containing calculated normalization results

`nds` Normalyzer dataset representing run data

`ner` Normalyzer evaluation results for running extended normalizations

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
emptyNormResults <- NormalyzerResults(normObj)
```

NormalyzerStatistics *Class representing a dataset for statistical processing in NormalyzerDE*

Description

Is initialized with an annotation matrix, a data matrix and a design data frame. This object can subsequently be processed to generate statistical values and in turn used to write a full matrix with additional statistical information as well as a graphical report of the comparisons.

Usage

```
NormalyzerStatistics(experimentObj, logTrans = FALSE)

NormalyzerStatistics(experimentObj, logTrans = FALSE)
```

Arguments

experimentObj	Instance of SummarizedExperiment containing matrix and design information as column data
logTrans	Whether the input data should be log transformed

Value

nds Generated NormalyzerStatistics instance

Slots

`annotMat` Matrix containing annotation information

`dataMat` Matrix containing (normalized) expression data

`filteredDataMat` Filtered matrix with low-count rows removed

`designDf` Data frame containing design conditions

`filteringContrast` Vector showing which entries are filtered (due to low count)

`pairwiseCompsP` List with P-values for pairwise comparisons

`pairwiseCompsFdr` List with FDR-values for pairwise comparisons

`pairwiseCompsAve` List with average expression values

`pairwiseCompsFold` List with log2 fold-change values for pairwise comparisons

`contrasts` Spot for saving vector of last used contrasts

`condCol` Column containing last used conditions

`batchCol` Column containing last used batch conditions

Examples

```
data(example_stat_summarized_experiment)
nst <- NormalyzerStatistics(example_stat_summarized_experiment)
```

`normMethods` *Perform normalizations on Normalyzer dataset*

Description

Perform normalizations on Normalyzer dataset

Usage

```
normMethods(
  nds,
  forceAll = FALSE,
  normalizeRetentionTime = TRUE,
  quiet = FALSE,
  rtStepSizeMinutes = 1,
  rtWindowMinCount = 100,
  rtWindowShifts = 1,
  rtWindowMergeMethod = "mean",
  noLogTransform = FALSE
)
```

Arguments

nds	Normalyzer dataset object.
forceAll	Force all methods to run despite not qualifying for thresholds.
normalizeRetentionTime	Perform retention time based normalization methods.
quiet	Prevent diagnostic output
rtStepSizeMinutes	Retention time normalization window size.
rtWindowMinCount	Minimum number of datapoints in each retention-time segment.
rtWindowShifts	Number of layered retention time normalized windows.
rtWindowMergeMethod	Merge approach for layered retention time windows.
noLogTransform	Per default NormalyzerDE performs a log-transformation on the input data. If not needed, specify this option

Value

Returns Normalyzer results object with performed analyzes assigned as attributes

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
normResults <- normMethods(normObj)
```

---

performCyclicLoessNormalization

Cyclic Loess normalization

Description

Log2 transformed data is normalized by Loess method using the function "normalizeCyclicLoess". Further information is available for the function "normalizeCyclicLoess" in the Limma package.

Usage

```
performCyclicLoessNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- performCyclicLoessNormalization(example_data_only_values)
```

`performGlobalRLRNormalization` *Global linear regression normalization*

Description

Log2 transformed data is normalized by robust linear regression using the function "rlm" from the MASS package.

Usage

```
performGlobalRLRNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

`rawMatrix`	Target matrix to be normalized
`noLogTransform`	Assumes no need for log transformation

Value

Normalized matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- performGlobalRLRNormalization(example_data_only_values)
```

performQuantileNormalization *Quantile normalization is performed by the function "normalize.quantiles" from the package preprocessCore.*

Description

It makes the assumption that the data in different samples should originate from an identical distribution. It does this by generating a reference distribution and then scaling the other samples accordingly.

Usage

```
performQuantileNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- performQuantileNormalization(example_data_only_values)
```

performSMADNormalization *Median absolute deviation normalization Normalization subtracts the median and divides the data by the median absolute deviation (MAD).*

Description

Median absolute deviation normalization Normalization subtracts the median and divides the data by the median absolute deviation (MAD).

Usage

```
performSMADNormalization(rawMatrix, noLogTransform = FALSE)
```

Arguments

rawMatrix	Target matrix to be normalized
noLogTransform	Assumes no need for log transformation

Value

Normalized matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- performSMADNormalization(example_data_only_values)
```

---

performVSNNormalization

Log2 transformed data is normalized using the function "justvsn" from the VSN package.

Description

The VSN (Variance Stabilizing Normalization) attempts to transform the data in such a way that the variance remains nearly constant over the intensity spectrum

Usage

```
performVSNNormalization(rawMatrix)
```

Arguments

rawMatrix	Target matrix to be normalized

Value

Normalized matrix

Examples

```
data(example_data_only_values_small)
normMatrix <- performVSNNormalization(example_data_only_values)
```

`reduceTechnicalReplicates` *Remove technical replicates from data and design*

Description

Collapses sample values into their average. If only one value is present due to NA-values in other technical replicates, then that value is used.

Usage

```
reduceTechnicalReplicates(se, techRepColName, sampleColName)
```

Arguments

`se`	Summarized experiment where the assay contains the data to be reduced, and the colData the data frame
`techRepColName`	Technical replicates column name in colData
`sampleColName`	Sample names column name in colData

Details

Takes a SummarizedExperiment where the data is present as the assay and the colData contains the design conditions. In the design conditions there should be one column with the technical replicate groups and one column containing the sample names

Value

reducedSe Summarized experiment with reduced data

Examples

```
testData <- as.matrix(data.frame(
    c(1,1,1),
    c(1,2,1),
    c(7,7,7),
    c(7,9,7)))
colnames(testData) <- c("a1", "a2", "b1", "b2")
designDf <- data.frame(
    sample=c("a1", "a2", "b1", "b2"),
    techrep=c("a", "a", "b", "b"))
se <- SummarizedExperiment::SummarizedExperiment(
    assay=testData,
    colData=designDf
)
statObj <- reduceTechnicalReplicates(se, "techrep", "sample")
```

setupJobDir *Create empty directory for run*

Description

Creates a directory at provided path named to the jobname.

Usage

```
setupJobDir(jobName, outputDir)
```

Arguments

jobName	Name of the run.
outputDir	Path to directory where to create the output directory.

Value

Path to newly created directory.

Examples

```
setupJobDir("job_name", "path/to/outdir")
```

setupRawContrastObject *Prepare SummarizedExperiment object for statistics data*

Description

Prepare SummarizedExperiment object for statistics data

Usage

```
setupRawContrastObject(dataPath, designPath, sampleColName)
```

Arguments

dataPath	Path to raw data matrix
designPath	Path to design matrix
sampleColName	Name for column in design matrix containing sample names

Value

experimentObj Prepared instance of SummarizedExperiment

Examples

```
data_path <- system.file(package="NormalyzerDE", "extdata", "tiny_data.tsv")
design_path <- system.file(package="NormalyzerDE", "extdata", "tiny_design.tsv")
sumExpObj <- setupRawContrastObject(data_path, design_path, "sample")
```

`setupRawDataObject` *Prepare SummarizedExperiment object for raw data to be normalized containing data, design and annotation information*

Description

Prepare SummarizedExperiment object for raw data to be normalized containing data, design and annotation information

Usage

```
setupRawDataObject(
  dataPath,
  designPath,
  inputFormat = "default",
  zeroToNA = FALSE,
  sampleColName = "sample",
  groupColName = "group"
)
```

Arguments

`dataPath`	File path to data matrix.
`designPath`	File path to design matrix.
`inputFormat`	Type of matrix for data, can be either 'default', 'proteios', 'maxquantprot' or 'maxquantpep'
`zeroToNA`	If TRUE zeroes in the data is automatically converted to NA values
`sampleColName`	Column name for column containing sample names
`groupColName`	Column name for column containing condition levels

Value

experimentObj SummarizedExperiment object loaded with the data

Examples

```
data_path <- system.file(package="NormalyzerDE", "extdata", "tiny_data.tsv")
design_path <- system.file(package="NormalyzerDE", "extdata", "tiny_design.tsv")
df <- setupRawDataObject(data_path, design_path)
```

writeNormalizedDatasets *Write normalization matrices to file*

Description

Outputs each of the normalized datasets to the specified directory.

Usage

```
writeNormalizedDatasets(
  nr,
  jobdir,
  includePairwiseComparisons = FALSE,
  includeCvCol = FALSE,
  includeAnovaP = FALSE,
  normSuffix = "-normalized.txt",
  rawdataName = "submitted_rawdata.txt"
)
```

Arguments

nr	Results object.
jobdir	Path to output directory.
includePairwiseComparisons	Include p-values for pairwise comparisons.
includeCvCol	Include CV column in output.
includeAnovaP	Include ANOVA p-value in output.
normSuffix	String used to name output together with normalization names.
rawdataName	Name of output raw data file.

Value

None

Examples

```
data(example_summarized_experiment)
normObj <- getVerifiedNormalyzerObject("job_name", example_summarized_experiment)
normResults <- normMethods(normObj)
normResultsWithEval <- analyzeNormalizations(normResults)
outputDir <- tempdir()
writeNormalizedDatasets(normResultsWithEval, outputDir)
```

Index